The preceding pages of this thesis have dealt with different applications of optimization algorithms in structural design of lens systems. The main thrust was on the exploration and development of global or quasiglobal optimization procedures for the same. Results of our investigations presented herein amply demonstrate the inherent potential of the new algorithms in exploring promising solutions as sought. In order to carry out a validation study within a limited time frame, we confined ourselves only to the case of lenses with spherical interfaces. Nevertheless, the obvious advantages associated with our approach in the treatment of structural design of lens systems with diffractive/binary elements, gradient index lens, aspherics etc. should not be overlooked.

Our treatment of structural design of lens systems assumes thin lens components and makes use of primary aberration analysis.
Notwithstanding these limitations, our approach can provide useful results in most practical applications. However, we are currently exploring ways and means for at least reducing, if not totally overcoming these restrictions. The passage from a thin lens to a thick lens is well understood in the domain of primary aberrations, and the stop shift formulae provide a convenient linkage of the overall primary aberration targets with the central primary aberrations of the constituent components. No such analytical correspondence exists for finite aberrations. Indeed, treatments based on Hamilton’s characteristic function or the Hamilton-Brun’s eikonal are likely to be more useful in such ventures (Buchdahl, 1970; Velzel, 1988; Walther, 1995). Some preliminary works on these and related aspects have already been done, though not included in this thesis.

Finally, a few observations on the global optimization techniques should be in order. Optimization problems in lens design provide a veritable test ground for GO algorithms. As effectiveness of the latter depends to a large extent on the nature of the configuration space, further studies on case specific modifications of these algorithms are called for. Hopefully, these studies would lead to the emergence of powerful optimization techniques whose realm of application goes far beyond the scope of optical design in emerging areas of science and technology.
A FORTRAN CODE FOR DAMPED LEAST SQUARES (DLS) ALGORITHM

The code is written and compiled with the help of Microsoft FORTRAN 77 version 5.0. The program for cemented doublets uses two files, i) dls.for and ii) cemdob.for and the program for broken contact doublets uses two files, i) dls.for and ii) bcdob.for

All the three programs are given below:

FILE: dls.for

The file contains a main program and following subroutines:

INPUT: for specifying input data to the program, all the input data is read from the data file idls.dat

CONSTS: for specifying values for various constants, e.g., the constants related to penalty function, the vertices for the glass triangle

MCALC: for calculating merit functions for the doublet, uses two other subroutines; ABERS and PENFUN

ABERS: for calculating primary aberrations for the doublet

PENFUN: for calculating penalty functions

WEIGHT: for implementing dynamic weighting strategy, different sets of weights are specified here

GDAMPS: for adjusting the value of global damping factor

SOLVE: for solving simultaneous linear equations using Gaussian elimination with pivotal condensation

SEARCH: for implementing a crude line search technique

COMPARE: for comparing the obtained aberration values with the corresponding targets specified by the user

AUR: for calculating aperture utilization ratio

All program output is directed towards the standard output device i.e., displayed on the monitor.
C main program-implementation of the DLS algorithm
C optimum choice of glass combination from the given glass triangle
C for a cemented/broken contact doublet with given specifications using
C dls method for optimization.

DIMENSION X(7),YM(2),XQ(7),DX(21),S(101),XM(7),DXD(21),SA(3)
REAL K(2),KD,L,P(2),M(5,7),MT(7,5),K1(2),XDD(7),TD(21,21),
DX(21),MD(5,7),S11(7),S12(7),JD,IDD,AD(21,21),BD(21),V(21)
,ZN(7)
CHARACTER*1 Z
CHARACTER*20 DTYPE
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /ACOM/DTYPE,/BCOM/F1(5),F(3,2),Y(2),SIS(7),SI7(7),k
CALL INPUT(X,DELTA)
c calculation of constants
Q1=H*(D**2)*(KD**2)/(L)
Q2=(D**4)*(KD**3)/(L)
Q3=(D**2)*KD/(L)
CALL Consts(SID,SMD,KDD,JD,DD,DELTA,ij)
SIC=SID
65 ND=0
DD1=0.
IDD=0
BETA=1.
CALL MCA,L(X,K,Y,F1,F,P,PD,SM)
WRITE(*,11)
11 FORMAT(10X,TOTAL SPH. AB,COMA,CHR.AB (IN LAMBDA)/)
SA(1)=F1(1)*Q2+SAT(1)
SA(2)=F1(2)*Q1+SAT(2)
SA(3)=F1(3)*Q3+SAT(3)
WRITE(*,12)SA(1),SA(2),SA(3)
12 FORMAT(5X,F10.5)
IF (PD.NE.1.)THEN
WRITE(*,14)
14 FORMAT(10X,'VALUES OF PENALTY FUNCS')
WRITE(*,15)P(1),P(2)
15 FORMAT(5X,'P1=',E9.3,5X,'P2=',E9.3)
ELSE
WRITE(*,'Glasses -outside boundary')
ENDIF
13 WRITE(*,'STARTING VALUE OF MERIT FUNCTION')
WRITE(*,'SM')
WRITE(*,'23')
23 FORMAT(2X,'CORRESPONDING VALUES OF VARIABLES GIVEN IN'/
1 'FOLLOWING ORDER - SHAPE FACTORS(X1,X2),REFRACTIVE INDICES'/
1 '(N1,N2),NORMALIZED POWER OF THE 1st COMPONENT(K1),'/
1 'DISPERSIONS (DN1,DN2)'/)
WRITE(*,*)X(I),I=1,7
CALL AUR(X,D,K,KD,RQ)
WRITE(*,24)
24 FORMAT(10X,'APERTURE UTILIZATION RATIO',/)
WRITE(*,*)RQ
CALL COMPARE (SAT,SA,II)
IF (II.EQ.1) THEN
   II=II+1
   WRITE(*,*)'POSSIBLE SOLUTION -',II
   WRITE(*,*)CONTINUE ITERATION ?(y/n)
   READ(*,'(A)')Z
   IF (Z.NE.'y') STOP
   ELSE IF ((SM.LE.1.E-07).OR.(IT.EQ.1)) THEN
      WRITE(*,*)THE POINT OF CLOSEST APPROACH
      WRITE(*,*)XN(1),XN(2),XN(3)
      CALL MCALC(XN,K1,YM,F1,F,P,PD,SD)
      WRITE(*,19)
      19 FORMAT(10X,'TOTAL SPH. AB,COMA,CHR.AB (IN LAMBDA)/')
      SA(1)=F1(1)*Q2+SAT(1)
      SA(2)=F1(2)*Q1+SAT(2)
      SA(3)=F1(3)*Q3+SAT(3)
      WRITE(*,12)SA(1),SA(2),SA(3)
      WRITE(*,*)'MERIT FUNCTION1
      WRITE(*,*)F1
      STOP 'FURTHER IMPROVEMENT IMPOSSIBLE'
   ENDIF
   NDD=0
   NCOUNT=0
16   NCOUNT=NCOUNT+1
   CALL WEIGHT(W,F1,P,PD,IDD,ND,JD,SA,SAT,DD,DD1)
   WRITE(*,*)W.Fs',W(1),W(2),W(3)
   WRITE(*,*)ND',ND
   18   IF (DTYPE .EQ.'CEMENTED') THEN
      CALL CEMDOB(X,DX,N,M,MT,MD,V,TD,AD,BD,*20)
      WRITE(*,30)
      30 FORMAT(10X,'SOLUTION VECTOR - DELTA X')
      WRITE(*,*)DX(1),DX(2),DX(3)
   ELSE
      CALL BCDLOB(X,DX,N,M,MT,MD,V,TD,*20)
      WRITE(*,31)
      31 FORMAT(10X,'SOLUTION VECTOR - DELTA X')
      WRITE(*,*)DX(I),I=1,7
   ENDF
   GO TO 22
20   NDD=NDD+1
   IF (ND.EQ.1) GO TO 38
   GO TO 16
22   DO 25 I=1,7
      XM(I)=X(I)+DX(I)
25   CONTINUE
   CALL MCALC(XM,K1,YM,F1,F,P,PD,SD)
   WRITE(*,*)'FINAL VALUE OF MERIT FUNCTION'
189
WRITE(*,*) SD
  SQ = SD
32  IF (PD.EQ.1.) THEN
    WRITE(*,36)
36    FORMAT(2X,'One or both of the glasses are outside the boundary')
    WRITE(*,*) 'DAMPING FACTOR', SI3
    CALL MCALC(X,K,Y,F1,F,P,PD,SM)
    NDD = NDD + 1
    WRITE(*,*) ND, NDD, NCOUNT, ND, NDD, NCOUNT
38  IF ((ND .EQ. 1).AND.(NDD.EQ.NCOUNT)) THEN
    IG = 0
    SI3 = SIC
    CALL GDAMPS(DX, SI3, SID, IG, R, DELTA, IT)
    WRITE(*,*) 'GDAMPS ', SB
    SIC = SI3
    GO TO 65
ELSE IF ((ND.EQ.1).AND.(NDD.LT.NCOUNT)) THEN
    KDD = 1
    GO TO 65
ELSE
    GO TO 16
ENDIF
ENDIF
40  BETA = BETA * DELTA
45  DO 45 I = 1,7
    XM(I) = X(I) + BETA * DX(I)
47  CONTINUE
    SMD = SD
    WRITE(*,*) 'M.Fs'
    CALL MCALC(XM, K1, YM, F1, F, P, PD, SD)
    WRITE(*,*) SMD, SD
    IF ((PD.EQ.1.) OR (SD.GT.SMD)) GO TO 50
    GO TO 40
50  DO 47 I = 1,7
    XM(I) = X(I) + (BETA / DELTA) * DX(I)
47  CONTINUE
    CALL SEARCH(XM, DX, S, SD, DXD, F1, K, XQ, F, P, XDD, BETA, DELTA, R, PD, ID)
    WRITE(*,*) BETA, SI3, DELTA
    WRITE(*,*) R, R
33  IF ((SQ.NE.SD).OR.(ND.EQ.1).OR.(ND.EQ.0).AND.(KDD.EQ.1))) GO TO 34
1  GO TO 34
    BETA = 1.
    GO TO 16
34  IF (KDD.EQ.1) KDD = 0
    SI3 = SID
    IG = 1
    CALL GDAMPS(DXD, SI3, SID, IG, R, DELTA, IT)
    WRITE(*,*) 'DAMPING FACTOR', SI3
    WRITE(*,*) ND, ND
IF(SM.LT.SD) THEN
  DO 70 I=1,7
    XN(I)=X(I)
  70 CONTINUE
ELSE
  DO 75 I=1,7
    XN(I)=XDD(I)
  75 CONTINUE
ENDIF
DO 35 J=1,7
  X(J)=XDD(J)
35 CONTINUE
55 GOTO 10
60 END

SUBROUTINE INPUT(X,DELTA)
C THIS Subroutine LISTS ALL THE DATA WHICH ARE TO BE SPECIFIED
C at the input
REAL X(7),KD,L
CHARACTER*20 DTYPE
COMMON X(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /ACOM/DTYPE
OPEN (19,FILE='idls.dat',ERR=100,STATUS='OLD')
WRITE(*,2)
2 FORMAT(1X,'TYPE OF THE DOUBLET(BROKEN CONTACT/CEMENTED):')
READ(*,'(A)') DTYPE
WRITE(*,5)
5 FORMAT(10X,'SYSTEM PARAMETERS',/
WRITE(*,*)'POWER OF THE SYSTEM','HEIGHT OF P.M.R'
READ(19,*)KD,D
WRITE(*,*)KD,D
WRITE(*,*)'CONJUGATE VARIABLE','LAGRANGE INVARIANT'
READ(19,*)YD,H
WRITE(*,*) YD,H
WRITE(*,10)
10 FORMAT(10X,'DESIRED ABERRATION TARGETS(IN LAMBDA)/')
WRITE(*,'(SPH. ABR.,COMA, CHR. ABR. TARGETS')
READ(19,*)SAT(1),SAT(2),SAT(3)
WRITE(*,*)SAT(1),SAT(2),SAT(3)
WRITE(*,15)
15 FORMAT(10X,'INITIAL VALUES OF THE VARIABLES/')
IF(DTYPE.EQ.'CEMENTED') THEN
  WRITE(*,'SHAPE VARIABLE FOR THE FIRST COMP')
  READ(19,*)X(1)
  WRITE(*,*)X(1)
ELSE

191
WRITE(*,*) 'SHAPE VARIABLE FOR THE TWO COMPONENTS'
READ(19,*)X(1),X(2)
WRITE(*,*)X(1),X(2)
ENDIF
WRITE(*,*) 'REFRACTIVE INDICES OF THE TWO GLASSES'
READ(19,*)X(3),X(4)
WRITE(*,*)X(3),X(4)
WRITE(*,*) 'VALUES OF DISPERSION FOR TWO GLASSES'
READ(19,*)X(6),X(7)
WRITE(*,*)X(6),X(7)
WRITE(*,*) 'NORMALIZED POWER OF THE 1ST COMP.'
READ(19,*)X(5)
WRITE(*,*)X(5)
CLOSE(19)
SI3=1.
WRITE(*,*) 'PRESS 1-TO SPECIFY GLOBAL DAMPING FACTOR'
READ(*,*)U
IF (U.NE.1.) GOTO 18
WRITE(*,*) 'GLOBAL DAMPING FACTOR'
READ(*,*)SI3
18 DO 22 J=1,7
XI(J)=0.0
22 CONTINUE
GO TO 23
C WRITE(*,*) 'ENTER 1 TO SPECIFY TARGET VALUES FOR VARIABLE
C 1 CHANGES'
C READ(*,*)U1
C IF (U1.NE.1.) GO TO 23
C WRITE(*,20)
C 20 FORMAT(10X,'DESIRED VALUES OF DELTA X(X1,X2,N1,N2,K1,DN1,DN2)'),
C READ(*,*)(XI(J),J=2,7)
100 STOP 'idls.dat file does not exist.'
23 RETURN
END

SUBROUTINE CONSTS(SID,SMD,KDD,JD,DD,DELTA,IJ)
REAL JD,KD,L
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,H,SI3,KD,L,Q1,Q2,Q3

C constants for penalty function calculations
AL=1.0
BE=.03
GA=30.0
C vertices of glass triangle
X1(1)=.0024
X1(2)=0.051
X1(3)=0.006
Y1(1)=1.36
Y1(2)=1.92
Y1(3)=1.85
X1(4)=X1(1)
X1(5)=X1(2)
Y1(4)=Y1(1)
Y1(5)=Y1(2)
c step length beta
DELTA=1.3
SID=SI3
c initialization of certain constants
SMD=0.
KDD=0
JD=.0001
DD=1000.
IJ=0
return
end

SUBROUTINE COMPARE (SAT,SA,II)
REAL SAT(3),SA(3)
J=0
II=0
DO 5 I=1,3
   IF((ABS(SAT(I)-SA(I))).LE.5.)J=J+1
5 CONTINUE
IF (J.EQ.3) H=1
RETURN
END

SUBROUTINE AUR(X,D,K,KD,RQ)
C THIS SUBROUTINE CALCULATES MAXIMUM APERTURE UTILISATION RATIO
REAL X(7),K(2),KD,CD(4)
DO 5 I=1,2
   CD(I)=KD*K(I)*(X(I)>1.)/(2.*(X(I+2)-1.))
   CD(I+2)=KD*K(I)*(X(I)<1.)/(2.*(X(I+2)-1.))
5 CONTINUE
CDD=ABS(CD(1))
DO 10 I=1,3
   IF (CDD.GE.ABS(CD(I+1))) GO TO 7
   CDD=ABS(CD(I+1))
   GO TO 10
7 CDD=ABS(CD(T))
10 CONTINUE
RQ=CDD*D
RETURN
END

SUBROUTINE ABERS(F,Y,X,K)
C This subroutine calculates the total spherical aberration,total C central coma,total chromatic aberration for the given doublet
REAL S1(2),S2(2),CL(2),F(5),X(7),Y(2),K(2),KD,L
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
W(3),AI(2),BI(2),CI(2),DI(2),GI(2),YD,H,D,SI3,KD,L,Q1,Q2,Q3

DO 10 J=1,2
A1(J)=(X(J+2)+2.)/(X(J+2)*(X(J+2)-1)**2)
B1(J)=2.*((X(J+2)**2-1.)/(X(J+2)+2.)
C1(J)=(X(J+2)**2)/((X(J+2)-1.)**2)
D1(J)=X(J+2)/(X(J+2)+2.)
E1(J)=(X(J+2)+1.)/(X(J+2)**2)*(X(J+2)-1.)
G1(J)=2.*((X(J+2)+1.)/X(J+2)
S1(J)=(1./32.)*((K(J)**3)*(A1(J)**3)+((X(J)-B1(J)*Y(J)**2)+C1(J)
S2(J)=(1./4.)*((K(J)**2)*((E1(J)*X(J)-G1(J)*Y(J))
CL(J)=(K(J)/2.]*X(J+5)/(X(J+2)-1.)
10 CONTINUE
F(1)=S1(1)+S1(2)-SAT(1)/Q2
F(2)=S2(1)+S2(2)-SAT(2)/Q1
F(3)=CL(1)+CL(2)-SAT(3)/Q3
RETURN
END

SUBROUTINE PENFUN(X,F,P,PD)
C THIS SUBROUTINE CALCULATES THE PENALTY FUNCS FOR THE GLASSES
DIMENSION X(7),P(2),F(3,2)
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
W(3),AI(2),BI(2),CI(2),DI(2),GI(2),YD,H,D,SI3,KD,L,Q1,Q2,Q3

PD=0.
SUM=0.
DO 5 K=1,3
SUM=SUM+X1(K)*((Y1(K+2)-Y1(K+1))
5 CONTINUE
DO 10 J=1,3
A(J)=2.*((Y1(J+1)-Y1(J+2))/SUM
B(J)=-2.*((X1(J+1)-X1(J+2))/SUM
C(J)=((Y1(J)*X1(J+2)-Y1(J+1)+X1(J+1)*(Y1(J)+Y1(J+2))-X1(J+2)*
1 (Y1(J+1)))/SUM
10 CONTINUE
F(J,J)=(A(J)*X(J+5)+B(J)*X(J+2)+C(J)**2
IF (F(J,J),GE.1.1) GO TO 25
10 CONTINUE
P(1)=0.
P(2)=0.
DO 20 J=1,3
DO 20 I=1,2
P(I)=P(I)+(AL*F(J,I)+BE)**GA
20 CONTINUE
GO TO 30
25 PD=1.
}

194
SUBROUTINE MCALC(X,K,Y,F1,F,P,PD,S)
C CALCULATIONS RELATED TO THE MERIT FUNCTION, WHICH ARE TO BE
C REPEATED FREQUENTLY.
REAL X(7),F1(5),Y(2),P(2),F(3,2),L,KD,K(2)
CHARACTER*20 DTYPE
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,SI3,KD,L,Q1,Q2,Q3
COMMON /ACOM/DTYPE
K(1)=X(5)
K(2)=1.-K(I)
Y(1)=(YD+K(1)-1.0)/K(1)
Y(2)=(YD+K(1))/(1.0-K(1))
IF (DTYPE.EQ.'CEMENTED') THEN
X(2)=(K(1)/K(2))*((X(4>1.)/(X(3)-1.))*(X(1>1.)-1.
ENDIF
CALL ABERS(F1,Y,X,K)
CALL PENFUN(X,F,P,PD)
F1(4)=P(I)
F1(5)=P(2)
S=((F1(I))**2+((F1(2)))**2+((F1(3))*100.)*2
RETURN
END

SUBROUTINE WEIGHT(W,F1,P,PD,IDD,ND,JD,SA,SAT,DD,DD1)
C MERIT FUNCTION WEIGHTS ARE DYNAMICALLY VARIED
REAL W(3),F1(5),JD,SA(3),SAT(3),K(3),P(2),JC
IDD=IDD+1
IF (((DD1.EQ.1.).OR.(IDD.GE.7))GO TO 15
DC=DD/(10.**IDD)
IF ((PD.EQ.L).OR.((P(1).LE.10.).AND.(P(2).LE.10.)))GO TO 5
DC=0.01
DD1=1.
IDD=6
5 DO 10 I=1,3
K(I)=ABS(SA(I)-SAT(I))
W(I)=DC*ABS(K(I))
10 CONTINUE
RETURN
15 JC=JD*(10.**(IDD-6))
IF (JC.GT.100.) GO TO 25
DO 20 I=1,3
W(I)=JC
20 CONTINUE
RETURN
25 DO 30 I=1,3
K(I)=ABS(SA(I)-SAT(I))
W(I)=0.
IF ((P(1).GT.20.).OR.(P(2).GT.20.)) W(I)=.000001
IF((P(1).GT.10.).OR.(P(2).GT.10.)) W(I)=.0001
IF (K(I).GT.2.) W(I)=100.
30 CONTINUE
ND=1
50 RETURN
END

SUBROUTINE SOLVE(N,A,B,X,*
C Solves set of N simultaneous equations of form AX=B
C by pivotal condensation method, leaving A,B undisturbed.
REAL A(21,21),B(21),C(21,22),X(21),Y(21),P,T,AP
C Arbitrary limit on N imposed by above array dimensions.
200 FORMAT(/l 5X,'PIVOT OF,I2,'TH.IS LT E-20V20X,'SUBROUTINE
1 SOLVE IS ABANDONED')
M=N+1
DO 10 I=1,N
C(I,M)=B(I)
DO 10 J=1,N
C(I,J)=A(I,J)
A,B arrays now telescoped into C
DO 20 I=1,N
P=C(I,I)
I1=I+1
AP=ABS(P)
IF(I-N)110,120,120
110 DO 30 J=I1,N
IF(AP-ABS(C(J,I)))50,30,30
50 DO 60 K=1,M
C Swap Ith and Jth rows of C
T=C(I,K)
C(I,K)=C(J,K)
60 C(J,K)=T
P=C(I,I)
AP=ABS(P)
30 CONTINUE
C Ith eq.PIVOT is now in Ith col--
C arranged by swapping amongst lower eqs.
120 IF(ABS(P)-1.0E-50)150,140,140
150 WRITE(*,150)
140 RETURN 1
140 DO 80 KK=I1,M
K=I+M-KK
80 C(I,K)=C(I,K)/P
C Divide thru by PIVOT IN Ith equation.
IF(I-N)130,20,20
130 DO 80 J=I1,N
DO 80 KK=I1,M
K=I+M-KK
80  C(J,K)=C(J,K)-C(I,K)*C(J,I)
C Subtract (coflft of X(I) in Jth eq)*(Ith eq) from Jth eq. for J=I+1
C to N—This eliminates X(I) in all eqs. below Ith
20 CONTINUE
C Eqs now condensed into upper triangular set
C Soln vector now obtained by back substitution.
Y(N)=C(N,M)
N1=N-1
DO 90 II=1,N1
  I=I+N1-II
  Y(I)=C(I,M)
  I1=I+1
  DO 90 KK=I1,N
    K=I1+N-KK
90  Y(I)=Y(I)-C(I,K)*Y(K)
DO 100 I=1,N
100 X(I)=Y(I)
RETURN
END

SUBROUTINE SEARCH(X,DX,S,SD,DXD,F1,K,XQ,F,P,XDD,BETA,
1 DELTA,R,PD,ID)
C THIS SUBROUTINE USES A LINE SEARCH TECHNIQUE
DIMENSION P1(102),S(101),DXD(21),F1(5),P(2),X(7),DX(21)
REAL K(2),K1(2),KD,L,YM(2),XQ(7),F(3,2),XDD(7),XM(7),DX1(7)
COMMON X1(6),Y1(6),A(3),B(3),E(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S3,KD,L,Q1,Q2,Q3
P1(1)=0.
ID=1
WRITE(*, *)'MERIT FUNCTIONS'
DO 101 I=1,101
  P1(I+1)=P1(I)+.01
  DO 12 J=1,7
    DX1(J)=(BETA-BETA/DELTA)*DX(J)
12 CONTINUE
CALL MCALC(XM,K1,YM,F1,F,P,PD,SQ)
S(I)=SQ
WRITE(*, *)S(I)
SD=S(ID)
IF ((PD.EQ.1).OR.(S(I).GT.SD)) GO TO 15
ID=I
10 CONTINUE
SD=S(ID)
DO 25 J=1,7
  WRITE(*, *)P1(ID),DX1(J)
  DXD(J)=P1(ID)*DX1(J)
  XDD(J)=X(J)+DXD(J)
  XQ(J)=XDD(J)
25 CONTINUE
197
CONTINUE
RD=SD/S(1)
R=(1.-P1(ID))*BETA/DELTA+BETA*P1(ID)

WRITE(*,*)'RELATIVE VALUE OF MERIT FUNCTION'
WRITE(*,*)RD
WRITE(*,*)'MINIMUM VALUE OF MERIT FUNCTION'
WRITE(*,*)SD
WRITE(*,*)'CORRESPONDING VALUES OF VARIABLES'
WRITE(*,*)(XQ(J), J=1,7)
WRITE(*,*)'INCREMENT IN VARIABLES IN LINE SEARCH'
WRITE(*,*)(DXD(J), J=1,7)
DO 35 J=1,7
   DXD(J)=P1(ID)*DX1(J)+(BETA/DELTA)*DX(J)
35 CONTINUE
WRITE(*,*)(DXD(J), J=1,7)
RETURN
END

SUBROUTINE GDAMPS(DX3,SI3,SID,IG,R,DELTA,IT)
c dynamically adjusts the global damping factor to control
c the length of the solution vector
REAL DX3(21),k(2)
CHARACTER*20 DTYPE
COMMON /ACOM/DTYPE, /BCOM/F1(5),F(3,2),Y(2),SIS(7),SI(7),k
JC=0
IT=0
IF(DTYPE.EQ.'CEMENTED') THEN
   DX3(2)=DX3(1)
ENDIF
DO 10 I=1,7
   IF((0.LT.(ABS(DX3(I))).AND.((ABS(DX3(I))).LE.1.E-06))
      JC=JC+1
10 CONTINUE
IF((IG.EQ.0).and.(JC.EQ.7)) IT=1
IF (IG.EQ.1) THEN
   IF (JC.EQ.7) SI3=SI3/DELTA
ELSE
   SI3=SI3*DELTA
ENDIF
RETURN
END

*****************************************************************************************
FILE: cemdob.for
*****************************************************************************************

The file contains following subroutines:
CEMDOB: a problem specific subroutine, acts as an interface between the dls.for file and all the other subroutines (CDERIVES, CTRANSP, CDERIV2, CDFACS, CPRODMX) in cemdob.for file.

CDERIVS: for calculating the first-order derivatives of aberrations with respect to the design parameters for the cemented doublet using corresponding analytical expressions

CTRANSP: for determining the elements of the transpose of the first-order derivative matrix obtained by CDERIVS

CDERIV2: for calculating the second-order derivatives of aberrations with respect to the design parameters for the cemented doublet using corresponding analytical expressions

CDFACS: for calculating the complete damping factor, using the scale factors and the second derivatives for the cemented doublet

CPRODMX: for calculating the elements of the product matrix of the transpose and the first-order derivative matrix

```
SUBROUTINE eemdob(x,dx,n,m,mt,md,v,td,ad,bd,*)
REAL k(2),dx(21),m(5,7),md(5,7),mt(7,5),kd,l,x(7),bd(21),
1 ad(21,21),v(21),td(21,21)
COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /BCOM/F1(5),F(3,2),Y(2),S1S(7),S1(7),k
CALL CDERIVS(M,X)
CALL CTRANSP(M,MT)
CALL CDERIV2(MD,X)
19 CALL CDFACS(M,MD)
CALL CPRODMX(M,MT,V,TD)
DO 20 I=1,6
   BD(I)=V(I+1)
   DO 20 J=1,6
   AD(I,J)=TD(I+1,J+1)
20 CONTINUE
   N=6
   CALL SOLVE(N,AD,BD,DX,*25)
   DO 22 I=1,5
      DX(8-I)=DX(7-I)
22 CONTINUE
   DX(2)=0.0
   GO TO 35
25 RETURN 1
35 RETURN
END
SUBROUTINE CDERIVS(M,X)
```
THIS SUBROUTINE CALCULATES THE DERIVATIVES OF THE ABERRATION FUNCTIONS.

REAL X(7), R1(2), M(5,7), K(2), KD, L, R2(2), R3(2), S(2), SD(2)
COMMON XI(6), Y1(6), A(3), B(3), E1(2), AL, BE, GA, X(21), C(3), SAT(3),
1 W(3), A1(2), B1(2), C1(2), D1(2), G1(2), YD, H, D, SI3, KD, I, Q1, Q2, Q3
COMMON /BCOM/F1(5), F(3,2), Y(2), SIS(7), SI(7), k

M(4,7) = 0.
M(5,3) = 0.
M(3,5) = W(3) * (X(6)/(X(3)-1.) - X(7)/(X(4)-1.))
SUM3 = 0.
SUM4 = 0.
SUM5 = 0.
DO 10 I = 1, 2
M(3, I+5) = W(3) * K(I)/(X(I+2)-1.)
SUM3 = 0.
SUM4 = 0.
SUM5 = 0.
DO 10 I = 1, 2
M(3, I+2) = -W(3) * K(I)
SD(I) = W(2) * E1(I) * (K(I)**2)/2.
M(4, I+3) = 0.
M(5, I) = 0.
M(5, I+4) = 0.
S(I) = (W(1)/4.) * 2. * (K(I)**3) * (A1(I)* (X(I)-B1(I)*Y(I))
M(1, I+5) = 0.
M(2, I+5) = 0.
M(3, I) = 0.
M(4, I) = 0.
GD = GA-1.
SUM1 = 0.
SUM2 = 0.
DO 15 J = 1, 3
SUM1 = 2. * GA * AL * (B(J)*(AL*F(J,I)+BE)**GD)*SQRT(F(J,I))+SUM1
SUM2 = 2. * GA * AL * (A(J)*(AL*F(J,I)+BE)**GD)*SQRT(F(J,I))+SUM2
15 CONTINUE
M(I+3, I+2) = SUM1
M(I+3, I+5) = SUM2
M(2, I+2) = (W(2)/2.) * (K(I)**2) * (X(I)*(1.-X(I+2)**2-2.*X(I+2))
1/((X(I+2)-1.)*1.2))
R1(I) = -2. * (X(I+2)**2+3.*X(I+2)-1.)/(X(I+2)**2)/(X(I+2)-1.)***3
R2(I) = -4. * Y(I)* (X(I+2)**2+4.*X(I+2)+1.)/(X(I+2)+2.)***2
R3(I) = -2. * (X(I+2)**2+3.*X(I+2)-1.)/(X(I+2)+2.)***2
M(1, I+2) = (W(1)/4.)* (K(I)**3)* (R1(I)* (X(I)-B1(I)*Y(I))**2
1+ A1(I)* R2(I)* (X(I)-B1(I)*Y(I))-R3(I)
SUM3 = (W(2)/2.)* K(I)* (E1(I)* X(I)**2-G1(I)* (Y(I)+1.))-SUM3
SUM4 = (W(1)/4.)* (K(I)**2)* A1(I)* (X(I)-B1(I)*Y(I))**3.*Y(I)
1- B1(I)* (Y(I)+2.))-SUM4
SUM5 = (W(1)/4.)* (K(I)**2)**3.* C1(I)* D1(I)* Y(I)* (Y(I)+2.))+SUM5
10 CONTINUE
M(2, 5) = SUM3
M(1, 5) = SUM4 + SUM5
T = (X(2)+1.)/(X(1)-L)
M(1, 1) = S(1) + T*S(2)
M(2,1)=SD(1)+T*SD(2)
DO 30 J=1,5
M(J,2)=M(J,1)
M(J,1)=0.
30 CONTINUE

WRITE(*,20)
20 FORMAT(10X,'DERIVATIVE MATRIX'/)
WRITE(*,25)((M(J,I),I=1,7),J=1,5)
25 FORMAT(7(1X,E9.3)/)
RETURN

SUBROUTINE CTRANSP(M,MT)
C THIS SUBROUTINE CALCULATES THE TRANSPOSE OF THE DERIVATIVE
C MATRIX.
REAL M(5,7),MT(7,5)
DO 20 I=2,7
DO 20 J=1,5
MT(I,J)=M(J,I)
20 CONTINUE
WRITE(*,25)
25 FORMAT(10X,'TRANSPOSE OF THE DERIVATIVE MATRIX'/)
WRITE(*,30)((MT(I,J),J=1,5),I=2,7)
30 FORMAT(5(2X,E9.3)/)
RETURN

SUBROUTINE CPRODMX(M,MT,V,TD)
REAL M(5,7),MT(7,5),T(7,7),V(21),TD(21,21),
1 VD(7),P1(5),DX2(7),k(2),KD,L
COMMON XI(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,SIS(7),SI(7),k
P1(1)=W(1)*F1(1)
P1(2)=W(2)*F1(2)
P1(3)=W(3)*F1(3)*100.
P1(4)=F1(4)
P1(5)=F1(5)
DO 10 I=2,7
DX2(I)=SIS(I)*XI(I)
DO 10 J=2,7
TD(I,J)=0.
VD(I)=0.
DO 12 IK=1,5
TD(I,J)=TD(I,J)+MT(I,IK)*M(IK,J)
VD(I)=VD(I)+MT(I,IK)*P1(IK)
12 CONTINUE
T(I,J)=TD(I,J)
IF (I.NE.J) GO TO 15
TD(I,J)=TD(I,J)+SIS(I)
V(I) = -V(I) + DX2(I)
CONTINUE
WRITE(*,25)
WRITE(*,30)((T(I,J),J=2,7),I=2,7)
WRITE(*,35)
WRITE(*,40)((TD(I,J),J=2,7),I=2,7)
WRITE(*,42)
WRITE(*,45)(V(I),I=2,7)
RETURN
END

SUBROUTINE CDERTV2(MD,X)
C This subroutine calculates the second derivatives of the aberration functions.
REAL MD(5,7),K(2),P(2),P2(2),P3(2),P4(2),P5(2),P6(2),S1(2),
X(7),S2(2),S3(2),kd,l
COMMON X(6),Y(6),A(3),B(3),E1(2),AL,BE,GA,X1(21),C(3),SAT(3),
W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /BCOM/F1(5),F(3,2),Y(2),SIS(7),SI(7),k
GD=GA-2.
SUM1=0.
SUM4=0.
SUM5=0.
MD(5,3)=0.
MD(4,7)=0.
MD(5,5)=0.
DO 10 I=1,2
SUM2=0.
SUM3=0.
MD(2,I)=0.
DO 15 J=1,3
MD(J,1+5)=0.
MD(J+2,1)=0.
SUM2=SUM2+(A(J)**2)*((AL*F(J,1)+BE)**GD)*(AL*GA*F(J,1)+BE)*
2.*AL*GA
1
SUM3=SUM3+(B(J)**2)*((AL*F(J,1)+BE)**GD)*(AL*GA*F(J,1)+BE)*
2.*AL*GA
1
CONTINUE
MD(I+3,1+4)=0.
MD(I+3,1+2)=SUM3
MD(I+3,1+5)=SUM2
MD(1,1)=0.5*(K(I)**3)*A1(I)
SUM1=SUM1+E1(I)*X(I)-G1(I)

MD(3,I+2)=2.*K(I)*X(I+5)/((X(I+2)-1)**3)

SUM4=SUM4+K(I)*((A1(I)**(6.*(X(I)**2))-4.*B1(I)*X(I)*Y(I)+12.*(B1(I)**2))-2.*D1(I)+6.*C1(I))

SUM5=K(I)*((A1(I)**(8.*B1(I)*X(I)-4.*(B1(I)**2))*Y(I))+1)

4.*D1(I)*Y(I))-SUM5

P1(I)=(X(I)-B1(I)*Y(I))

P2(I)=(X(I+2)**2+4.*X(I+2)+1.)/(X(I+2)+2.)**2)

P3(I)=1./((X(I+2)-1.)**2)

P4(I)=1./(X(I+2)+2.)

P5(I)=(X(I+2)**2+3.*X(I+2)-1.)/(X(I+2)-1.)**3)

P6(I)=(2.*X(I+2)+3.)/(X(I+2)**2*(X(I+2)-1.)**3)

MD(2,I+2)=-(K(I)**2)*((P3(I)**2)*X(I)*(1.+E1(I)/X(I+2))/X(I+2)

+Y(I)/(X(I+2)**3))

WRITE(*,*)(P1(I),P2(I),P3(I),P4(I),P5(I),P6(I)

S3(I)=(P1(I)**2)*P6(I)-P5(I)*5.*X(I+2)-2.)/X(I+2))

S2(I)=-4.*A1(I)*(Y(I)**2)*(P2(I)**2)+2.*(P4(I)**3)*Y(I)*6.*

S1(I)=-2.*Y(I)*P3(I)+5.*(P1(I))/P5(I)*P3(I)*(1.+3.*X(I+2))

10 CONTINUE

MD(2,5)=SUM1+2.*G1(2)

MD(1,5)=(SUM4+SUM5)/4.

DO 30 J=1,5

MD(J,2)=MD(J,1)

MD(J,1)=0.

30 CONTINUE

WRITE(*,20)

FORMAT(10X,'SECOND DERIVATIVES')

WRITE(*,25)(MD(J,J),J=1,7)

FORMAT(7(1X,E9.3))

RETURN

END

SUBROUTINE CDFACS(M,MD)

C This subroutine calculates the damping factors following

C Hopkins method using the values of second derivatives.

REAL DS1(7),DS2(7),S1(7),S2(7),M(5,7),MD(5,7),S1(7),k(2)

1 ,KD,L

COMMON X1(6),Y1(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),

W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3

COMMON /BCOM/F1(5),F2(5),F3(3),Y(2),Y2(5),SIS(7),S1(7),k

S12(1)=.01

S12(2)=.01

S12(3)=.05

S12(4)=.05

S12(5)=.01

S12(6)=.005

S12(7)=.005

203
DO 10 I=2,7
SUM1=0.
SUM2=0.
DO 15 J=1,5
DO 15 IK=2,7
SUM1=SUM1+2.*F1(J)*M(J,I)*SI2(I)+M(J,I)*M(J,IK)*SI2(I)*SI2(I)
SUM2=SUM2+F1(J)*MD(J,I)*(SI2(I)**2)
15 CONTINUE
DS1(I)=SUM1
DS2(I)=SUM2
S1(I)=DS2(I)/DS1(I)
S1(I)=1.+ABS(S1(I))
SI(I)=(SI1(I)/SI2(I))*SI3
SIS(I)=SI(I)*SI(I)
10 CONTINUE
C WRITE(*,20)
C 20 FORMAT(10X,'DAMPING FACTORS/)
C WRITE(*,25)(SI(I),I=2,7)
C 25 FORMAT(3X,E9.3)
RETURN
END

******************************************************************************
FILE: bcdob.for
******************************************************************************

The file contains following subroutines

BCDOB: a problem specific subroutine, acts as an interface between the dls.for file and all the other subroutines (BDERIVES, BTRANS, BDERIV2, BDFACS, BPRODMX) in bcdob.for file.

BDERIVES: for calculating the first-order derivatives of aberrations with respect to the design parameters for the broken contact doublet using corresponding analytical expressions

BTRANS: for determining the elements of the transpose of the first-order derivative matrix obtained by

BDERIV2: for calculating the second-order derivatives of aberrations with respect to the design parameters for the broken contact doublet using corresponding analytical expressions

BDFACS: for calculating the complete damping factor, using the scale factors and the second derivatives for the broken contact doublet

BPRODMX: for calculating the elements of the product matrix of the transpose and the first-order derivative matrix
SUBROUTINE BCDOB (x,dx,n,m,mt,md,v,td,*)
REAL k(2),dx(2l),m(5,7),md(5,7),mt(7,5),kd,l,x(7)
,TD(21,21),V(21)
COMMON XI(6),YI(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),AI(2),BI(2),Cl(2),DI(2),GI(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /BCOM/F1(5),F3(2),Y(2),SIS(7),SI(7),k

18 CALL BDERIVS(M,X)
CALL BTRANS(M,MT)
CALL BDERIV2(MD,X)
19 CALL BDFACS(M,MD)
CALL BPRODMX(M,MT,V,TD)
N=7
CALL SOLVE(N,TD,V,DX,*25)
GO TO 30
25 RETURN 1
30 RETURN
END

SUBROUTINE BDERIVS(M,X)
C THIS SUBROUTINE CALCULATES THE DERIVATIVES OF THE ABERRATION
C FUNCTIONS.
REAL M(5,7),K(2),KD,L,R2(2),R3(2),X(7),RI(2)
COMMON XI(6),YI(6),A(3),B(3),E1(2),AL,BE,GA,XI(21),C(3),SAT(3),
1 W(3),AI(2),BI(2),Cl(2),DI(2),GI(2),YD,H,D,S13,KD,L,Q1,Q2,Q3
COMMON /BCOM/F1(5),F3(2),Y(2),SIS(7),SI(7),k

M(4,7)=0.
M(5,3)=0.
M(3,3)=W(3)*(X(6)/(X(3)-1.)-X(7)/(X(4)-1.))
SUM3=0.
SUM4=0.
SUM5=0.
DO 10 I=1,2
M(3,I+5)=W(3)*K(I)/(X(I+5)-1.)
M(3,I+2)=W(3)*K(I)*X(I+5)/(X(I+2)-1.)**2
M(2,I)=W(2)*E1(I)*K(I)**2/2
M(4,I+3)=0.
M(5,1)=0.
M(5,I+4)=0.
M(1,I+5)=W(1)/4.*2.*(K(I)**3)*(AI(1)*(X(I)-BI(1)*Y(I)))
M(1,I+5)=0.
M(2,I+5)=0.
M(3,I)=0.
M(4,I)=0.
GD=GA-1.
SUM1=0.
SUM2=0.
205
DO 15 J=1,3
SUM1=2.*GA*AL*(B(JD*(AL*F(J,I)+BE)**GD)*SQRT(F(J,I))+SUM1
SUM2=2.*GA*AL*(A(J)*(AL*F(J,I)+BE)**GD)*SQRT(F(J,I))+SUM2
15 CONTINUE
M(I+3,I+2)=SUM1
M(I+3,I+5)=SUM2
M(2,I+2)=(W(2)/2.)*(K(I)**2)*W(I-X(I+2)**2-2.*X(I+2))
R1(I)=-2.*(X(I+2)**2+3.*X(I+2)-1.)/(X(I+2)**2)/(X(I+2)-1.)*3
R2(I)=-4.*Y(I)*(X(I+2)**2+4.*X(I+2)+1.)/(X(I+2)+2.)*2**2
R3(I)=2.*(X(I+2)/((X(I+2)-1.)*3)-Y(I)**2)/((X(I+2)+2.)*2**2))
M(I,1+2)=(W(I)/4.)*(K(I)**2)*(R1(I)*(X(I)-B1(I)*Y(I))-2**2)
SUM3=(W(2)/2.)*K(I)*(E1(I)*X(I)**2.-G1(I)*(Y(I)+1.)) -SUM3
SUM4=(W(I)/4.)*(K(I)**2)*A1(I)*(X(I)-B1(I)*Y(I))*(3.*X(I))
-1.*B1(I)*(Y(I)+2.)) -SUM4
SUM5=(W(I)/4.)*(K(I)**2)*((C1(I)-D1(I)*Y(I))*(Y(I)+2.))+SUM5
10 CONTINUE
M(2,5)=-SUM3
M(1,5)=-SUM4+SUM5
WRITE(*,20)
20 FORMAT(10X,'DERIVATIVE MATRIX'/)
WRITE(*,25)((M(J,I),I=1,7), J=1,5)
25 FORMAT(7(1X,E9.3)/)
RETURN
END

SUBROUTINE BTRANSP(M,MT)
C THIS SUBROUTINE CALCULATES THE TRANSPOSE OF THE DERIVATIVE
C MATRIX.
REAL M(5,7),MT(7,5)
DO 20 I=1,7
DO 20 J=1,5
MT(I,J)=M(I,J)
20 CONTINUE
WRITE(*,25)
25 FORMAT(10X,'TRANSPOSE OF THE DERIVATIVE MATRIX'/)
WRITE(*,30)((MT(I,J),J=1,5),I=1,7)
30 FORMAT(7(1X,E9.3)/)
RETURN
END

SUBROUTINE BPRODMX(M,MT,V,TD)
REAL M(5,7),MT(7,5),T(7,7),V(21),TD(21,21), VD(7),
P1(5),DX2(7),k(2),KD,L
COMMON X1(6), Y1(6), A(3), B(3), E1(2), AL, BE, GA, XI(21), C(3), SAT(3),
206


```plaintext
1 W(3),A1(2),B1(2),C1(2),D1(2),G1(2),YD,H,D,S13,KD,L,Q1,Q2,Q3

COMMON /BOM/Fl(5),F(3,2),Y(2),SIS(7),SI(7),k

P1(1)=W(1)*F1(1)
P1(2)=W(2)*F1(2)
P1(3)=W(3)*F1(3)*100.
P1(4)=F1(4)
P1(5)=F1(5)

DO 10 I=1,7
   DX2(I)=SIS(I)*XI(I)

DO 10 J=1,7
   TD(I,J)=0.
   VD(I)=0.

   DO 12 IK=1,5
      TD(I,J)=TD(I,J)+MT(I,IK)*M(IK,J)
      VD(I)=VD(I)+MT(I,IK)*P1(IK)
   CONTINUE

   T(I,J)=TD(I,J)
   IF (I.NE.J) GO TO 15
   TD(I,J)=TD(I,J)+SIS(I)
15 V(I)=-VD(I)+DX2(I)

10 CONTINUE

C WRITE(*,25)
   FORMAT(10X,'PRODUCT MATRIX'/)
C WRITE(*,30)((T(I,J), J= 1,7),I= 1,7)
C WRITE(*,35)
   FORMAT(10X,'COEFFICIENT MATRIX',/)
C WRITE(*,40)((TD(I,J), J= 1,7),I= 1,7)
C WRITE(*,42)
   FORMAT(10X,'COLUMN OF THE CONSTANTS'/)
C WRITE(*,45)(V(I),I=1,7)
C WRITE(5X,E9.3)
C WRITE(*,47)(VD(I),I=1,7)
   WRITE(*,48)(DX2(I),I=1,7)
C WRITE(*,49)(P1(I),I=1,5)
RETURN
END

SUBROUTINE BDERIV2(MD,X)
C This subroutine calculates the second derivatives of the
C aberration functions.
DIMENSION X(7),S1(2),S2(2),S3(2)
REAL MD(5,7),K(2),P1(2),P2(2),P3(2),P4(2),P5(2),P6(2),kd,l
```

207
COMMON /BCOM/F1(5), F(3,2), Y(2), S1(7), SI(7), k

GD = GA - 2.
SUM1 = 0.
SUM4 = 0.
SUM5 = 0.
MD(5, 3) = 0.
MD(4, 7) = 0.
MD(5, 5) = 0.
DO 10 I = 1, 2
SUM2 = 0.
SUM3 = 0.
MD(2, I) = 0.
DO 15 J = 1, 3
MD(J, I + 5) = 0.
MD(J + 2, I) = 0.
SUM2 = SUM2 + (A(J) ** 2) * ((AL * F(J, I) + BE)**GD) * (AL * GA * F(J, I) + BE) * 2 * AL * GA
SUM3 = SUM3 + (B(J) ** 2) * ((AL * F(J, I) + BE)**GD) * (AL * GA * F(J, I) + BE) * 2 * AL * GA
15 CONTINUE
MD(I + 3, I + 4) = 0.
MD(I + 3, I + 2) = SUM3
MD(I + 3, I + 5) = SUM2
MD(I, I) = 0.5 * (K(I) ** 3) * A1(I)
SUM1 = SUM1 + E1(I) * X(I) - G1(I)
MD(3, I + 2) = 2 * K(I) * X(I + 5) / (X(I + 2) - 1)**3
SUM4 = SUM4 + K(I) * (A1(I) * ((X(I) ** 2) - 4 * B1(I) * X(I) * Y(I)) + 2 * (B1(I) ** 2)) - 2 * D1(I) + 6 * C1(I))
SUM5 = K(I) * (A1(I) * (8 * B1(I) * X(I) - 4 * (B1(I) ** 2) * Y(I)) + 4 * D1(I) * Y(I) + 1)
14 * D1(I) * Y(I) - SUM5
P1(I) = (X(I) - B1(I) - Y(I))
P2(I) = (X(I) + 2)**2 + 4 * (X(I) + 2)**2 + 1. / ((X(I) + 2 + 2)**2)
P3(I) = 1. / (X(I) + 2 - 1.)
P4(I) = 1. / (X(I + 2) + 2.)
P5(I) = (X(I) + 2)**2 + 3 * (X(I) + 2) - 1. / ((X(I) + 2)**2 - 4 * (X(I) + 2)**3)
P6(I) = (X(I) + 2)**2 - 3. / ((X(I) + 2)**2 - 1.**3)
MD(2, I + 2) = -(K(I)**2) * (P3(I)**2) * X(I)**2 + 1. + E1(I) / X(I + 2) / X(I + 2)
+ Y(I) / (X(I + 2)**2))
14 WRITE(*, *) P1(I), P2(I), P3(I), P4(I), P5(I), P6(I)
S1(I) = (P1(I)**4) / (P2(I) - P3(I) + P4(I) * P5(I) - 5 * (X(I + 2)**2) / X(I + 2))
14 S1(I) = 2 * Y(I) * P5(I) * P1(I) * P2(I) + (P3(I)**2) * (1. - 3 * X(I + 2) - 1. * X(I + 2))
14 P3(I)
MD(1, I + 2) = -(K(I)**3) * (S1(I) + S2(I) + S3(I)) / 2.
10 CONTINUE
MD(2, 5) = SUM1 + 2 * G1(2)
SUBROUTINE BDFACS(M, MD)

C This subroutine calculates the damping factors following
C Hopkins method using the values of second derivatives.

REAL M(5,7), MD(5,7), S1(7), DS1(7), DS2(7), SI1(7), SI2(7), k(2)

COMMON X(6), Y(6), A(3), B(3), E1(2), AL, BE, GA, XI(21), C(3), SAT(3),
W(3), A1(2), B1(2), C1(2), D1(2), G1(2), YD, H, D, SI3, KD, L, Q1, Q2, Q3
COMMON /BCOM/ F1(5), F(3,2), Y(2), SIS(7), SI(7), k

SI2(1) = .01
SI2(2) = .01
SI2(3) = .05
SI2(4) = .05
SI2(5) = .01
SI2(6) = .005
SI2(7) = .005

DO 10 I = 1, 7
SUM1 = 0.
SUM2 = 0.
DO 15 J = 1, 5
DO 15 IK = 1, 7
SUM2 = SUM2 + F1(J) * MD(J,I) * (SI2(I) * SI2(I))
15 CONTINUE
DS1(I) = SUM1
DS2(I) = SUM2
S1(I) = DS2(I) / DS1(I)
SI1(I) = 1. + ABS(S1(I))
SI(I) = (SI1(I) * SI2(I)) * SI3
SIS(I) = SI(I) * SI(I)
10 CONTINUE

WRITE(*,20)
WRITE(*,25)(SI(I), I = 1, 7)
RETURN
END
B FORTRAN CODE FOR GSA ALGORITHM

The code is written in Microsoft FORTRAN 77 version 5.0. The complete code is segmented in four files and these files are to be compiled simultaneously to run the program.

Four files are i) gsamain.for ii) gsautil.for iii) dblsup.for iv) mfdoblet.for

Files gsamain.for and gsautil.for contain the main program and the subroutines required for the operation of the algorithm itself. The file mfdoblet.for calculates the merit function for the cemented or broken contact doublet. The file dblsup.for acts as an interface between the algorithm specific files (gsamain.for and gsautil.for) and the application specific file mfdoblet.for. Codes for all these four files are given below.

For the triplet problem the same gsa files will be used in conjunction with a support file for triplets (triplsup.for) and mftripl.for to calculate triplet merit function. The codes listed in Appendix D for these triplet files can only be used with gsa files with slight modifications for I/O list of parameters for the subroutines.

When the glass variables are discrete, the algorithm requires one additional file, the glass library glaslib.dat, the contents of which are given in Appendix E.

FILE: gsamain.for

This file contains the main program and the following subroutines:

initdata: it may be used either for accepting algorithm specific input data interactively from the terminal and writing it to a data file or for inputting data from a data file called igs.dat

anneal: for setting the annealing schedule

thermeq: for generating constrained random walk for the continuous as well as the discrete variables

shuffle: for implementing the shuffling algorithm to be used with generation of discrete variables

metrop: for determining the acceptance probability for a particular step

ran3: a function, generates random fractions

All the program output goes to the file 'sgaout1'. A part of the program output comes from the dblsup.for file.

PROGRAM GSA
Main program. Implements GSA algorithm for continuous and discrete variables. Uses subroutines anneal, thermq, metrop, globalm and function ran3. Subroutine initdata is used for algorithm specific data input,

1. Generation strategy for continuous variables:
A new strategy is adopted for random step generation, to ensure that the steps are generated within the physical boundary of the variables, hence the step generation is not wasteful. This generation technique is directly applicable for variables having simple rectangular boundaries. For variables having other than rectangular boundary (id1.ne.0)--- (e.g. glass variables for optics design problems, which move within a triangular space), we need to incorporate some additional tricks (range subroutine).

2. Generation strategy for discrete variables:
Exponential deviate is used to generate new glasses in the close neighborhood of glasses contained in previous solution- small step random walk, designed for glasses arranged in ascending order of their r.i. values (glaslib2.dat).

This file works with other three files:
- gsautil.for --contains subroutines range, penfun. These subroutines are designed for triangular glass boundaries. Normally glasses are treated as discreet variables, in which case these routines have no part to play. In general, if some of the problem variables have triangular search domains, they can use these routines.
- mf*.for --for calc. of problem specific merit function
- *sup.for --as support for the particular problem of choice. This support file may not be required always.

In our case, the merit function is mostly related to optics design problem. Output is reduced in order to make it fast running.

Modifications of original gsdbl.for file on 8 Aug, 1999.
Some variables are renamed.

x(20): running variable; r: step length; t: temperature; tfactr: factor by which temperature is lowered.

gsamain2.for and gsamain3.for(this file) are the same. Only this file is used to find out the local minima(glasses are shuffled once at the begining) whereas gsamain2.for is used to find the global minimum(shuffling is done at each iteration).
Algorithm specific Input is taken from igsa.dat file and application specific input from ibcdbl.dat(for bcdoublet) or
REAL*8 x(20),de,t,xd(20),xa(20),r,x1(6),y1(6),fitfunc,e,
1 xdumy(20),pdisp(2),tabug(2,100)
LOGICAL ans
CHARACTER*1 ch
COMMON de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta,/KCOM/ii,tabug
COMMON /CCOM/xl ,y 1 ,/JCOM/iflag,iarray(3,100)/HCOM/xdummy,pdisp
if (id1.ne.0) then
  (xl(i), y1(i)), i=1,3 are the preassigned coordinates for the
c  edges of the glass triangle; (xl(i),y1(i)), i=4,5 are taken
c  for facilitating cyclic interchange in penfun subroutine
  Glass triangle may be chosen otherwise, only while specifying
  the vertices here, one must be careful to maintain the following
  order for the values of y l(i), y 1(1)=lowest value of r.i.,
c  y1(2)=highest value of r.i.
  
  x1(1) = 0.0024
  x1(2) = 0.051
  x1(3) = 0.006
  x1(4) = x1(1)
  x1(5) = x1(2)
  y1(1) = 1.36
  y1(2) = 1.92
  y1(3) = 1.85
  y1(4) = y1(1)
  y1(5) = y1(2)
c  x1(1) = 0.001
c  x1(2) = 0.06
c  x1(3) = 0.002
c  x1(4) = x1(1)
c  x1(5) = x1(2)
c  y1(1) = 1.2
  y1(2) = 2.0
  y1(3) = 1.9
  y1(4) = y1(1)
c  y1(5) = y1(2)
endif
write(*,*)'How many times the program should be run?'
read(*,*)nrun
50 do ii=1,nrun
  call initdata(x)
  iflag=1
  write(*,*)'Is there any other problem related constants to be
  read from input data file?(y/n)'
c  read(*,'(A)')ch
  ch='y'
  if(ch.eq.'y')call input()
c  initial point is generated within the rigid boundary.
open(20,ACCESS='APPEND',file='gsaout1',status='UNKNOWN')
call GETTIM (ihr,imin,isec,i100th)
write(20,*)'Initial time',ihr,imin,isec,i100th
idum =-i100th
if (idum.eq.0) idum=-1
write(20,*)'seed',idum
c write(*,*)'seed',idum
c read(*,*)idum
do 1 i=1,nvar
 if(id1.ne.0.and.i.eq.id2) call range(xa)
   xa(i)=xd(i)+dr(i)*ran3(idum)
 if((id1.eq.0).and.(i.ge.id2).and.(id2.ne.0))then
   xa(i)=xd(i)+int(dr(i)*ran3(idum))
   do 2 j=1,1+dr(i)
     iarray(i-id2+1,j)=j+xd(i)-1
 2 continue
 endif
1 continue
if((id1.eq.0).and.(id2.ne.0))call shuffle(idum)
write(20,*)'initial point',(xa(i),i=1,nvar)
e=fitfunc(xa)
write(20,7)'x',(xdummy(i),i=1,7)
7 format(1x,a1,7f11.8//)
write(20,*)'meritf =',e

c PAUSE
c write(*,*)x(1),x(2),x(3),x(4),x(5),x(6),x(7)
c initial settings of constants
c initial estimation of temperature
c currently with the comments before statement no. 10 no initial
c temperature rescheduling is done; only acceptance ratio is noted
c for the given problem

do 15 i=1,10
call thermeq(idum,x,nsucc,nd,ans)
   ndsucc=ndsucc+nsucc
   ndd=ndd+nd
15 continue
 accra=(float(ndsucc))/(float(ndd))
c write(*,*)'steps tried:',ndd
c write(*,*),total accepted steps=',ndsucc,
c & 'acceptance ratio=',accra
if(accr.gt.0.4)go to 10

t=2.*t

write(*,*)'temp=',t
go to 5

write(20,*)'initial temperature',t

write(*,*)'no. of iteration 1'
iw=20

write(*,*)'Where do you want to see your output?'
1 (con/prn/filename)'
call anneal(iw,idum,x,nsucc,nd,ans)
nq=nq+nd
write(20,*)'xa',(xa(i),i=1,nvar)

write(iw,*)'Domain of search:'
write(iw,*)'Origin'
write(iw,26)(dr(i),i=1,nvar)
call applchar(iw,xa)
close(20)
do 45 ij=id2,nvar
   tabug(ij-id2+1,ii)=xa(ij)
45 continue
50 continue
29 end

SUBROUTINE initdata(x)

c input data
REAL*8 de,r,t,xd(20),xa(20),pdisp(2),xdummy(20),x(20)
CHARACTER*1 a
COMMON de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta,/BCOM/tfactr,
   /HCOM/xdummy,pdisp
   id2=0
   write(*,*)'Do you want to create the input data file?(Y/N)?'
read(*,*)a
   a='n'
   if ((a.eq.'Y').or.(a.eq.'y')) then
      open(19,file='new')
   write(*,*)'No. of independent variables'
   read(*,*)nvar
   write(19,*)nvar
   write(*,*)'No. of variables not having rectangular boundary'
      read(*,*)id1
   write(19,*)id1
   if (id1.ne.0)then
      write(*,*)'Our program can only handle a triangular boundary (other than rectangular domain)'
   write(*,*)'Position of the variable in the series'
      read(*,*)id2
   write(19,*)id2

214
else
    write(*,*)'If some of the variables are discrete, arrange
    them sequentially after all the continuous variables and
    specify the position from where discrete variables start.
    If no discrete variable is present, please enter 0 at this
    prompt.'
    read(*,*)id2
    write(19,*)id2
    if(id2.ne.0) then
        write(*,*)'Decay factor for the exponential deviate'
        read(*,*)beta
        write(19,*)beta
    endif
    write(*,*)'If there are any variables to remain fixed
    during a complete run of the algorithm, enter the number of such
    variables at this prompt, if no such variables, enter 0'
    read(*,*)id3
    write(19,*)id3
    if(id3.ne.0) then
        write(*,*)'Specify values of these variables'
        read(*,*)(xa(nvar+i),i=1,id3)
        write(19,*)(xa(nvar+i),i=1,id3)
        write(*,*),give partial dispersion values'
        read(*,*)(pdisp(i),i=1,2)
        write(19,*)(pdisp(i),i=1,2)
        do 35 i=1,id3
            x(nvar+i)=xa(nvar+i)
        35 continue
    endif
endif
write(*,*)'Origin'
read(*,*)(xd(i),i=1,nvar)
write(19,*)(xd(i),i=1,nvar)
write(*,*)'Maximum acceptable change in the variables'
read(*,*)(dr(i),i=1,nvar)
write(19,*)(dr(i),i=1,nvar)
write('Initial Temperature','step length'
read(*,*)t,r
write(19,*)t,r
write('Temperature reduction schedule'
read(*,*)tfactr
write(19,*)tfactr
write('Acceptance probability: power of the merit func'
read(*,*)m
write(19,*),m
else
    open(19,file='igsa.dat',err=99,status='old')
    write(*,*)No. of independent variables'
    read(19,*)nvar
    write(*,*)nvar
write(*,*)'No. of variables not having rectangular boundary'
read(19,*)id1
write(*,*)id1

if (id1.ne.0) then
  write(*,*)'Our program can only handle a triangular boundary (other than rectangular domain)'
  write(*,*)'Position of the variable in the series'
  read(19,*)id2
  write(*,*)id2
else
  write(*,*)'If some of the variables are discrete, arrange them sequentially after all the continuous variables and specify the position from where discrete variables start. If no discrete variable is present, please enter 0 at this prompt.'
  read(19,*)id2
  write(*,*)id2
  if (id2.ne.0) then
    write(*,*)'Decay factor for the exponential deviate'
    read(19,*)beta
    write(*,*)beta
  endif
  write(*,*)'If there are any variables to remain fixed during a complete run of the algorithm, enter the number of such variables at this prompt, if no such variables, enter 0'
  read(19,*)id3
  write(*,*)id3
  if (id3.ne.0) then
    write(*,*)'Specify values of these variables'
    read(19,*)xa(nvar+i), i=1,id3
    write(*,*)(xa(nvar+i), i=1,id3)
    write(*,*)'Give partial dispersion values'
    read(19,*)pdisp(i), i=1,2
    write(*,*)(pdisp(i), i=1,2)
    do 40 i=1,id3
      x(nvar+i)=xa(nvar+i)
      continue
  endif
endif

write(*,*)'Origin'
read(19,*)xd(i), i=1,nvar
write(*,*)(xd(i), i=1,nvar)
write(*,*)'Maximum acceptable change in the variables'
read(19,*)dr(i), i=1,nvar
write(*,*)(dr(i), i=1,nvar)
write(*,*)'Initial Temperature','step length'
read(19,*)t,r
write(*,*)t,r
write(*,*)'Temperature reduction schedule'
read(19,*)tfactr
write(*,*)tfactr
write(*,*)'Acceptance probability: power of the merit func'
read(19,*)m
write(*,*)m
close(19)
endif
go to 10
99 stop 'specified data file does not exist'
10 return
dend

SUBROUTINE anneal(iw,idum,x,nsucc,nd,ans)
c  Uses ran3,mfunc,thermeq
LOGICAL ans
REAL*8 x(20),de,t,r,xd(20),xa(20)
COMMON de,dr(20),xd,xa,nvar,idl,id2,r,m,t,beta,/BCOM/tnfactr
t=t*tfactr
ndd=0
c 20000 temperature steps
do 10 j=1,20000
   nsucc=0
call thermeq(idum,x,nsucc,nd,ans)
   ndd=ndd+nd
c  if(pd. eq. 1.) go to 20
C 2   write(*,5)TEMP=',t
C 5    format(1x,a5,e14.8)
C   write(*,*)'no. of iterations',j+1
C  write(*,*)'successful moves:',nsucc
C  write(*,*)'total steps :',nd
C  annealing schedule
   t=tfactr*t
   if no success, we are done
      if (nsucc.eq.0)go to 20
10  continue
20   write(iw,30)TEMP=',t
30    format(1x,a5,e14.8)
write(iw,*)'no. of iterations',j+1
write(iw,*)'successful moves:',nsucc
write(iw,*)'total steps :',nd
nd=ndd
SUBROUTINE thermeq(idum,x,nsucc,nd,ans)

describes the process leading to thermal equilibrium.
Uses ran3,fitfunc

REAL ran3,rand
REAL*8 x(20),x1(20),x2(20),xd(20),xa(20),de,r,t,r1,y1
1 ,fitfunc,e,ed,xdummy(20),pdisp(2),tabug(2,100)
LOGICAL ans
COMMON de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta,/HCOM/xdummy,pdisp
1 ,/JCOM/itlag,iarray(3,100),/KCOM/ii,tabug

c write(*,*)'**',x(l),x(2),x(3),x(4),x(5),x(6),x(7)
c nd: counter for steps undertaken;
c nsucc: no. of accepted detrimental steps
c nover: maximum possible no. of steps
c nlimit: upper limit for the no. of accepted detrimental steps

nd=0
nsucc=0
nover=50*nvar
nlimit=5*nvar
e=fitfunc(xa)
C write(*,*)'ultimate meritf =',e
C write(*,5)'xa',(xa(i),i=1,nvar)
C 5 format(1x,a2,5fl5.8)
C write(*,7)'x',(xdummy(i),i=1,7)
C 7 format(1x,a1,7fl1.8//)

c maximum possible length of Markov chain given by nover
do 14 k=1,nover
nd=nd+1

c random step generation with uniform generation probability.
c subroutine 'range' incorporates the triangular boundary for
c glass variables in the generation process.

sum=0.0
ncont=nvar
if((id1.eq.0).and.(id2.ne.0))ncont=id2-1
do 17 i=1,ncont
if((id1.ne.0).and.(i.eq.id2)) then
   call range (x)
endif
   x(i)=xd(i)+dr(i)*ran3(idum)
x1(i)=x(i)-xa(i)
   sum=sum+(x1(i)**2)
17 continue
   y1=sqrt(sum)
   return
end

SUBROUTINE thermeq(idum,x,nsucc,nd,ans)
c write(*,*)'yl',yl
r1=r
if(r.ge.yl) r1=yl
do 24 i=1,nvar
   x2(i)=x1(i)*r1/yl
   x(i)=xa(i)+x2(i)
24 continue
if(ncont.ne.nvar)then
   call shuffle(idum)
   indx=0
   do 26 i=id2,nvar
      do 27 j=l,dr(i)+l
         if (iarray(i-id2+1,j).eq.int(xa(i))) then
            ipick=j
            go to 28
         endif
      27 continue
      28 rand=ran3(idum)
         if (rand.eq.0.) go to 28
         yy=-log(rand)/beta
         if(rand.gt..5) then
            iexpdev=mod(int(yy),2+int(dr(i)))
         else
            iexpdev=-mod(int(yy),2+int(dr(i)))
         endif
         c write(*,*)'expdev',iexpdev
         c write(*,*)'ipick',ipick
         if((ipick+iexpdev).gt.(l+dr(i))) then
            j=mod(ipick+iexpdev-1-int(dr(i)),2+int(dr(i)))
         else if((ipick+iexpdev).lt.1) then
            j=mod(ipick+iexpdev+l+int(dr(i)),2+int(dr(i)))
         else
            j=ipick+iexpdev
         endif
         x(i)=iarray(i-id2+1,j)
   if(ii.ne.1) then
      do 29 ij=l,ii-l
         if(x(i).eq.tabug(i-id2+1,ij)) then
            indx=indx+l
            go to 30
         endif
      29 continue
      30 endif
   if(indx.eq.(nvar-id2+1)) then
      indx=indx-l
      go to 28
   endif
26 continue
219
write(*,*)'accepted step',(xa(i),i=1,nvar)
write(*,*),'random step',(x(i),i=1,nvar)

meritfunction calculation

ed=fitfunc(x)
write(*,*),'meritf with random step generation=',ed
write(*,*),'penalty indicator=',pd

change in merit function

de=(ed-e)/(e**m)
write(*,*),'change in meritf=',de
write(*,*),'last accepted meritf=',e
call metrop(de,t,ans)
if(ans)then
nsucc=nsucc+1
write(*,*),'accepted steps=',nsucc
endif
if(nsucc.lt.nd)beta=1.
if(nsucc.ge.nlimit) return
14 continue

return
end

subroutine shuffle(idum)
This routine shuffles the glass array
REAL ran3
REAL*8 xd(20),xa(20),de,r,t
common de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta,
1/JCOM/iflag,iarray(3,100)
do 10 i=id2,nvar
do 10 j=1,1+dr(i)
other=j+int((2+dr(i)-j)*ran3(idum))
temp=iarray(i-id2+1,other)
iarray(i-id2+1,other)=iarray(i-id2+1,j)
iarray(i-id2+1,j)=temp
10 continue
return
end
SUBROUTINE metrop(de,t,ans)
   c Uses ran3,Metropolis algorithm, ans is a logical variable
   c that issues a verdict on whether to accept a step that leads
   c to a change de in the merit function e, if de<0. ans=true,
   c while if de>0. ans is only true with probability exp(-de/t)
   c where t is the temperature determined by the annealing schedule

   REAL ran3
   REAL*8 de,t
   LOGICAL ans
   DATA jdum/1/
   c use of DATA statement instead of assignment statement for
   c initialization of jdum is important. This allows subsequent
   c changes in jdum during a single run of the main program.

   if(de.lt.0.0)then
      ans=.true.
   else
      ans=(ran3(jdum).lt. exp(-de/t))
   endif
   return
end

FUNCTION ran3(idum)
   c ran3 returns a uniform random deviate between 0.0 and 1.0.
   c A negative value is to be given for idum to initialize or
   c re-initialize the sequence.

   INTEGER MBIG,MSEED,MZ
   INTEGER mj,mk,ma(55)
   REAL MBIG,MSEED,MZ
   REAL mj,mk,ma(55),ran3
   c According to Knuth, any large MBIG, but any smaller (but still
   c large) MSEED can be substituted for the above values.

   SAVE iff,inext,inextp,ma
   DATA iff/0/
   c Note : see comments on the use of DATA statement in metrop
   c subroutine

   MBIG=4000000.
   MSEED=1618033.
   MZ=0.
   FAC=1./MBIG
c initialization or reinitialization

```fortran
if (idum.lt.0.or.iff.eq.0) then
  iff=1
end if
```

c initializing ma(55) using the seed idum and large no. MSEEK

```fortran
mj=MSEEK-abs(idum)
mj = mod(mj,MBIG)
ma(55)=mj
mk=1
do 10 i=1,54
  ii=mod(21*i,55)
  ma(ii)=mk
  mk=mj-mk
  if(mk.lt.MZ)mk=mk+MBIG
  mj=ma(ii)
10 continue
do 13 k=1,4
do 12 i=1,55
  ma(i)=ma(i)-ma(1+mod(i+30,55))
  if(ma(i).lt.MZ)ma(i)=ma(i)+MBIG
12 continue
13 continue
inext=0
inextp=31
idum=1
endif
inext=inext+1
if(inext.eq.56)inext=1
inextp=inextp+1
if(inextp.eq.56)inextp=1
mj=ma(inext)-ma(inextp)
if(mj.lt.MZ)mj=mj+MBIG
ma(inext)=mj
b=mj
ran3=b*FAC
return
end
```

FILE: gsautil.for

This file contains utilities for the main gsa algorithm. It contains following subroutines:

**range**: this subroutine is only needed, when glass variables are considered continuous. It is called by the subroutine thermeq to implement the triangular glass boundary at the generation stage.
**fitfunc**: designed to act as an interface between the problem specific subroutine for merit function calculation and the algorithm specific subroutines say *thermeq*

**SUBROUTINE range**

C This routine corresponds exclusively to a triangular boundary
C for the available glass domain

REAL*8 x(20),md(3),cd(3),de,t,xa(20),x1(6),y1(6),r,term(3)
COMMON de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta
COMMON /CCOM/X1,Y1

DO 5 I=1,3
    MD(I)=(Y1(I)-Y1(I+1))/(X1(I)-X1(I+1))
    CD(I)=Y1(I)-X1(I)*MD(I)
5 CONTINUE

DO 10 I=ID2,ID2+IDL-1
    DO 15 J=1,3
        TERM(J)=(X(I-2)-CD(J))/MD(J)
15 CONTINUE
    IF (X(I-2).GT.Y1(3)) TERM(3)=TERM(2)
    DR(I)=TERM(1)-TERM(3)
    XD(I)=TERM(3)
10 CONTINUE

RETURN
END

**FUNCTION fitfunc**

c to call subroutine objfunc
 REAL*8 meritf, fitfunc,x(20)
 CHARACTER*100 charstr
 COMMON /ECOM/meritf,charstr
 call objfunc(x)
 c write(*,*)'meritf,meritf'
 fitfunc=meritf
 return
END

***************************************************************************************************
**FILE:** dblsup.for
***************************************************************************************************

This file contains following subroutines:

applchar: to output optical characteristics for the optimum doublet structure

gconvert: to read the refractive index and dispersion value corresponding to the integer produced during the step generation process for discrete glass from the glaslib.dat file.
**input:** to input data specific to the doublet problem considered for our application from the data file igsadbl.dat. It can also be used for entering data interactively and writing the same in a data file

**sysconst:** calculates the constants to be applied to the normalized forms of primary aberrations to get the actual Seidel aberration values. These constants are calculated using the paraxial specifications for the doublet in question.

**objcfunc:** calls the subroutine for merit function calculation **mfunc**

c Designed for variation of merit
c function due to change in three primary abrrations only, for
c thin doublets, cemented and broken contact. In general, the
c variables are shape factors, power distribution and two glasses.
c Glass variables are allowed to take values from a wellchosen
c list of 64 glasses, specified in glasslib.dat file.
c Output is reduced in order to make it fast running.
c This file works with other three files : sgautil.for - contains
c utilities in the form of subprogram modules coding different
c portions of SGA and mfbcdbl.for --for calc. of bcdoublet
c meritfunction or mfcmdbl.for --for calc. of cemdoublet
c meritfunction also dblsup.for --as support for doublet problems.
c Input data are taken from two data files :- algorithm specific
c data from isga.dat, doublet specific data from isgadbldat.

C wl: wavelength; kt: total power of the doublet; x(20): running
C variable; sa(5): sum of aberrations; sat(5): sum of aberration
C targets; f(5): normalized aberrations and pseudoaberrations
C r: step length; q(5): normalizing factors; yt: conjugate
C variable for the doublet; pd: strong penalty violation index;
c Development started sometime in October 1998.
SUBROUTINE applchar(iw,x)
REAL*8 kt,x(20),meritf,sa(5),sat(5),f(5),y(2),k(2),q(5),yt
$ ,aurc,aurb,fitnes,fitfunc,xdummy(20),pdisp(2),w(5)
CHARACTER*100 charstr
CHARACTER*1 ch
COMMON /DCOM/ sat,kt,yt,/ECOM/meritf,charstr,/FCOM/id,f,y,k,q,
$ w,ch,/HCOM/xdummy,pdisp

fitnes= fitfunc(x)
indx=0
do 25 i=1,4
   sa(i)=f(i)*q(i)
   if(abs(sa(i)-sat(i)).le.10) indx=indx+1
25 continue
write(iw,*)'x',(xdummy(i),i=1,id)
if(indx.ge.3)then
   write(iw,*)'meritf is',charstr,'func. in',id,'dimensions.'
   if (ch.eq.'c') then
      c
Write(iw,*')merit is',charstr,func. in',id,'dimensions.'
if (ch.eq.'c') then

224
write(iw,*)'Cemented doublet problem --optimum values
variables are — 1. shape variable (XI) of the first
comp., 2. normalized power (k1 bar) of the first comp.,
3. r.i.(n1) 4. dispersion (del n1) of the first comp.,
5. r.i.(n2) 6. dispersion (del n2) of the second comp.'
else
write(iw,*)'Broken contact doublet problem —optimum values
variables are — shape variables 1. (XI) and 2. (X2) of
the 2 comp.s, 3. normalized power (k1 bar) of the first
comp., 4. r.i.(n1) 5. dispersion (del n1) of the first comp.,
6. r.i.(n2) 7. dispersion (del n2) of the second comp.'
endif
write(iw,20)(xdummy(i),i= 1 ,id)
20 format(lx,7fl 1.8)
write(iw,*)'Partial dispersion values for the two glasses'
write(iw,20)(pdisp(i),i= 1,2)
c if(pd.eq. 1.) write(*,*),warning: glasses may be unacceptable'
c write(iw,32)'f,qs',(f(i),i=l,4),(q(i),i=l,4)
c 32 format(lx,a5,6el4.8)
write(iw,*)'TotaI primary sph. abr. (S1/8), coma (S2c/2),
# chr. abr. (CL/2), secondary spectrum (SL) (in wavelengths)'
write(iw,41)(sa(i),i= 1,4)
41 format(5x,4fl0.5)
write(iw,*)'Corresponding meritf value ',meritf
write(iw,*)'Aperture Utilization Ratio'
if (ch.eq.'c') then
  write(iw,*)aurc(xdummy)
else
  write(iw,*)aurb(xdummy)
endif
endif
call GETTIM (ihr,imin,isec,i100th)
write(iw,*)'Final time',ihr,':',imin,':',isec,':',i100th
return
29 end

SUBROUTINE gconvert(x)
c glasses are picked from a list of 64 preferred glasses
REAL*8 x(20),ri(2),disp(2),pdisp(2),xdummy(20),xd(20),xa(20)
@ ,de,r,f(5),q(5),y(2),k(2),w(5)
CHARACTER*1 ch
COMMON de,dr(20),xd,xa,nvar,id1,id2,r,m,t,beta,/HCOM/xdummy,
@ pdisp,F COM/id,f,y,k,q,w,ch
c following three lines are for gsa operation with continuous
c variables only
do 1 i=1,nvar
  xdummy(i)=x(i)
1 continue
do 5 i=1,2
  ip=idnint(x(nvar+i-2))
OPEN(19, file='glaslib.dat', recl=17, form='FORMATTED')
c  write(*,*)'ip',ip
  do 2 j=1,ip-1
    read(19,*)
  2 continue
  read(19, (BN, f8.6, 1x, f8.6, 1x, f8.6)) ri(i), disp(i), pdisp(i)
c  write(*,*)'ri, disp, pdisp', ri(i), disp(i), pdisp(i)
c  continue
  do 10 i=1,2
    xdummy(id-4+i)=ri(i)
    xdummy(id-2+i)=disp(i)
  10 continue
  c  write(*,15)'x',(xdummy(i), i= 1, id)
c  15 format(1x, a, 7f12.8)
c  return
end
SUBROUTINE inputQ

c application specific input data
REAL*8 kt, wl, sat(5), HH, yt, h, w(5), f(5), y(2), k(2), q(5)
CHARACTER*1 a, ch
COMMON /GCOM/ wl, HH, h, /DCOM/ sat, kt, yt, /FCOM/id, f, y, k, q, w, ch
write(*,*)'Input for doublet problem (Cemented/Broken contact)'
c write(*,*)'Do you want to create the input data file? (Y/N)'
c read(*,'(A)') a
  a='n'
    if ((a.eq.'Y').or.(a.eq.'y')) then
      open(19, file='status-new')
      write(*,*)'Specify doublet type (c/b)'
      read(*,'(A)') ch
      write(19, ' (A)') ch
      write(*,*)'system constants : Lagrange invariant, height of pmr'
      read(*,*) HH, h
      write(19, *) HH, h
      write(*,*)'total power of the system, operating wavelength'
      read(*,*) kt, wl
      write(19, *) kt, wl
      write(*,*)'conjugate variable for the thin doublet'
      read(*,*) yt
      write(19, *) yt
      write(*,*)'aberration targets (sph. ab (S1/8)T, coma (S2c/2)T, @
          chr. ab (CL/2)T, secondary spectrum (SL)T )'
      read(*,*) (sat(i), i= 1, 4)
      write(19, *) (sat(i), i= 1, 4)
      write(*,*)'Weights for the merit function'
      read(*,*) (w(i), i= 1, 4)
      write(19, *) (w(i), i= 1, 4)
close(19)
  else
    226
OPEN(19, FILE='ibcdbl.daf', ERR=99, STATUS='OLD')
WRITE(*,*)'Specified doublet type (c/b)'
READ(19,'(A)') CH
WRITE(*,'(A)') CH
WRITE(*,*)'system constants : Lagrange invariant, height of pmr'
READ(19,*) HH, H
WRITE(*,*) HH, H
WRITE(*,*)'total power of the system, operating wavelength'
READ(19,*) KT, WL
WRITE(*,*) KT, WL
WRITE(*,*)'conjugate variable for the thin doublet'
READ(19,*) YT
WRITE(*,*) YT
WRITE(*,*)'aberration targets (sph. ab (S1/8)T, coma (S2c/2)T, @ chr. ab (CL/2)T, secondary spectrum (SL/T))'
READ(19,*)(SAT(I), I=1,4)
WRITE(*,*)(SAT(I), I=1,4)
WRITE(*,*)'weights for the merit function'
READ(19,*)(W(I), I=1,4)
WRITE(*,*)(W(I), I=1,4)
CLOSE(19)
END
SUBROUTINE SYSCONST()
C calculation of system constants
REAL*8 KT, WL, SAT(5), HH, YT, H, F(5), Y(2), K(2), Q(5), W(5)
CHARACTER*1 CH
COMMON /GCOM/ WL, HH, H, /DCOM/ SAT, KT, YT, /FCOM/ ID, F, Y, K, Q, W, CH
IF (CH.EQ.'B') ID=7
IF (CH.EQ.'C') ID=6
Q(1)=(H**4)*(KT**3)/(WL)
Q(2)=HH*(H**2)*(KT**2)/(WL)
Q(3)=(H**2)*K/(WL)
Q(4)=Q(3)
RETURN
END
SUBROUTINE OBJFUNC(X)
C works as mediator between functions MFUNC and FITFUNC
REAL*8 X(20)
C following line should be made a comment for gsa operation
C with continuous variables only
C call gconvert (X)
This file contains the following functions and subroutines:

- **aurb**: Calculates aperture utilization ratio for the broken contact doublet.
- **abersb**: Calculates the values of normalized primary aberrations for the broken contact doublet.
- **mfunc**: Calculates merit function for any of the two doublet types.
- **aurc**: Calculates aperture utilization ratio for the cemented doublet.
- **abersc**: Calculates the values of normalized primary aberrations for the cemented doublet.

### FUNCTION AURB(X)

```fortran
FUNCTION AURB(X)

C THIS SUBROUTINE CALCULATES MAXIMUM APERTURE UTILIZATION RATIO

REAL*8 X(20),K(2),KT,CD(4),f(5),y(2),wl,HH,q(5),w(5),
1 sat(5),yt,aurb

CHARACTER*1 ch

COMMON /DCOM/ sat,kt,yt,/FCOM/id,f,y,k,q,w,ch,/GCOM/ wl,HH,h

IN=1

DO 5 I=1,2
  CD(IN)=KT*K(I)*(X(I)+1)/(2*(X(I+3)-1))
  CD(IN+1)=KT*K(I)*(X(I)-1)/(2*(X(I+3)-1))
  IN=IN+2

5 CONTINUE

CDD=ABS(CD(1))

DO 10 I=1,3
  IF (CDD.GE.ABS(CD(I+1))) GO TO 7
  CDD=ABS(CD(I+1))
  GO TO 10

7 CDD=ABS(CD(I))

10 CONTINUE

aurb=CDD*h

RETURN

END
```

### SUBROUTINE ABERSB(X)

```fortran
SUBROUTINE ABERSB(X)

C This subroutine calculates the total spherical aberration, total
```

228
REAL*8 K(2), KT, S1(2), S2(2), CL(2), X(20), f(5), A1(2), B1(2), C1(2),
1 D1(2), E1(2), G1(2), y(2), yt, sat(5), q(5), xdummy(20), pdisp(2), w(5)
CHARACTER*1 ch
COMMON /DCOM/ sat, kt, yt, /FCOM/ id, f, y, k, q, w, ch, /HCOM/ xdummy, pdisp

DO 10 J=1,2
  A1(J)=(X(J+3)+2.)/(X(J+3)*(X(J+3)-1.)**2)
  B1(J)=2.*(X(J+3)**2-1.)/(X(J+3)+2.)
  C1(J)=(X(J+3)**2)/(X(J+3)**2)
  D1(J)=X(J+3)/(X(J+3)+2.)
  E1(J)=(X(J+3)+1.)/(X(J+3)**2)
  G1(J)=(2.*X(J+3)+1.)/X(J+3)
  S1(J)=(1./4.)*(K(J)**3)*(A1(J)*((X(J)-B1(J)*Y(J))**2)+C1(J)
1  -D1(J)*(Y(J)**2))
  S2(J)=(1./2.)*(K(J)**2)*(E1(J)*X(J)-G1(J)*Y(J))
  CL(J)=K(J)*X(J+5)/(X(J+3)-1.)
10 CONTINUE

F(1)=(S1(1)+S1(2))/8.
F(2)=(S2(1)+S2(2))/2.
F(3)=(CL(1)+CL(2))/2.
c write(*,*)'pdisp',pdisp(1),pdisp(2)
F(4)=(CL(1)*pdisp(1)+CL(2)*pdisp(2))
c write(*,*)'f(4)',f(4)
RETURN
END

SUBROUTINE mfimc(X)
C CALCULATIONS RELATED TO THE MERIT FUNCTION, WHICH ARE TO BE
C REPEATED FREQUENTLY.
REAL*8 KT, K(2), X(20), MERITF, q(5), xdummy(20), pdisp(2)
1 ,f(5), sat(5), y(2), YT, w(5)
  CHARACTER*100 charstr
CHARACTER*1 ch
COMMON /DCOM/ sat, kt, yt, /ECOM/ meritf, charstr, /FCOM/ id, f, y, k, q,
1 w, ch, /HCOM/ xdummy, pdisp
if(ch.ne.'c')then
  charstr ="OPTICAL MERIT FUNCTION FOR BROKEN CONTACT DOUBLET"
  K(1)=X(3)
else
  charstr ="OPTICAL MERIT FUNCTION FOR CEMENTED DOUBLET"
  K(1)=X(2)
endif
if (k(1).eq.0.0.or.k(1).eq.1.0)k(1)=k(1)+0.1
C Following three lines are needed when the glasses are fixed,
C optimization w.r.t. the continuous variables are sought
do 3 i=1,id-4
  xdummy(i)=x(i)
3 continue
K(2)=1.-K(1)

229
Y(1) = (YT + K(1) - 1.0) / K(1)
Y(2) = (YT + K(1)) / (1.0 - K(1))
if (ch.eq.'c') then
   CALL ABERSc(Xdummy)
else
   call abersb(xdummy)
endif
meritf = 0.
do 5 i = 1, 4
   MERITF = meritf + (w(i) * (q(i) * F(i) - sat(i)))**2
5 continue
RETURN
END

FUNCTION AURc(X)
C This subroutine calculates maximum aperture utilization ratio
REAL*8 X(20), K(2), KT, CD(4), f(5), w1, Hh, q(5), w(5)
1 , y(2), sat(5), XCEM(20), Yt, h, aurc
   CHARACTER*1 ch
COMMON /DCOM/ sat, kt, yt, /FCOM/ id, f, k, q, w, ch, /GCOM/ w1, Hh, h
COMMON /ICOM/ XCEM
   IN = 1
   DO 5 I = 1, 2
      CD(IN) = KT * K(I) * (XCEM(I) + 1.0) / (2.0 * (X(I + 2) - 1.0))
      CD(IN + 1) = KT * K(I) * (XCEM(I) - 1.0) / (2.0 * (X(I + 2) - 1.0))
      IN = IN + 2
5 CONTINUE
   CDD = ABS(CD(1))
   DO 10 I = 1, 3
      IF (CDD .GE. ABS(CD(I + 1))) GO TO 7
      CDD = ABS(CD(I + 1))
      GO TO 10
7 CDD = ABS(CD(I))
10 CONTINUE
   AURc = CDD * h
RETURN
END

SUBROUTINE ABERSc(X)
C This subroutine calculates the total spherical aberration, total coma, total chromatic aberration for the given air-spaced doublet
REAL*8 K(2), KT, S1(2), S2(2), CI(2), X(20), y(2), xcem(20), w(5)
1 , A1(2), B1(2), C1(2), D1(2), E1(2), G1(2), xdummy(20), pdisp(2),
1 f(5), sat(5), Q(5), yt
   CHARACTER*1 ch
COMMON /DCOM/ sat, kt, yt, /FCOM/ id, f, y, k, q, w, ch, /ICOM/ XCEM,
1 /HCOM/xdummy, pdisp
   write(*, *)'k(2), q(1), q(2), q(3)
   XCEM(1) = X(1)
   XCEM(2) = (K(1)/K(2)) * ((X(4) - 1.0) / (X(3) - 1.0)) * (X(1) - 1.0) - 1.
   DO 10 J = 1, 2
   ...
A1(J) = \frac{(X(J+2)+2.)(X(J+2)*X(J+2-1))}{(X(J+2)+2.)}

B1(J) = 2.\frac{(X(J+2)**2-1.)(X(J+2)+2.)}{(X(J+2)+2.)}

C1(J) = \frac{(X(J+2)**2)/((X(J+2)-1.)**2)}{(X(J+2)+2.)}

D1(J) = \frac{X(J+2)/(X(J+2)+20}{(X(J+2)+2.)}

E1(J) = \frac{(X(J+2)+1.)(X(J+2)*X(J+2-1.))}{(X(J+2)+2.)}

G1(J) = \frac{(2.*X(J+2)+1.)/(X(J+2))}{(X(J+2)+2.)}

S1(J) = \frac{(1./4.*K(J)**3)*(A1(J)*((XCEM(J)-B1(J)*Y(J)**2)+}{(X(J+2)+2.)}

S1(J) = \frac{(1./2.*K(J)**2)*(E1(J)*XCEM(J)-G1(J)*Y(J))}{(X(J+2)+2.)}

CL(J) = K(J)*X(J+4)/(X(J+2)-1.)

10 CONTINUE

F(1) = \frac{(S1(1)+S1(2))}{8.}

F(2) = \frac{(S2(1)+S2(2))}{2.}

F(3) = \frac{(CL(1)+CL(2))}{2.}

F(4) = CL(1)*pdisp(1)+CL(2)*pdisp(2)

WRITE(*,*)'f',f(i),i=1,4

RETURN

END
FORTRAN CODE FOR SGA ALGORITHM

The code is written in Microsoft FORTRAN 77 version 5.0. The complete code is segmented in two files i) sgamain.for ii) sgautil.for and these files are to be compiled simultaneously with the application specific files e.g., dblsup.for and mfdoblet.for to run the program for doublets and with triplsup.for and mftripl.for to run the program for triplets. The same dblsup.for and mfdoblet.for files listed in APPENDIX B can be used with slight modifications. To save space both the versions of the same files are not given. The codes for the triplet files are given in APPENDIX D.

Files sgamain.for and sgautil.for contain the main program and the subroutines required for the operation of the algorithm itself. Codes for these files are given here.

When the glass variables are discrete, the algorithm requires one additional file, the glass library glaslib.dat, the contents of which are given in APPENDIX E.

******************************************************************************

FILE: sgamain.for
******************************************************************************

This file contains the main program and the following subroutines:

initdata: it may be used either for accepting algorithm specific input data interactively from the terminal and writing it to a data file or for inputting data from a data file isga.dat

outdata: to write program output (statistics for various generations of populations except the starting generation) on the terminal (standard output device).

PROGRAM sga

main program for a simple genetic algorithm (SGA) version 1.
uses binary coding of problem variables. Ref. D.E. Goldberg
uses subroutine initdata for algorithm specific data input,
function DECODE for retrieving data from bit strings.

Output is reduced in order to make it fast running.
This file works with other three files:
sgautil.for - contains utilities in the form of subprogram
modules coding different portions of SGA
mf*.for --for calc. of problem specific merit function
*sup.for --as support for the particular problem of choice.
This support file may not be required always.
Input data are taken from two data files :- algorithm specific
data from isga.dat, problem specific data from isga*.dat.
where, * represents wild card character and stands for a set of
letters characterizing the specific problem.
c In our case, the merit function is mostly related to optics design
problem.

INTEGER gen,1(10),popsize
REAL*8 x(20),xd(20),dr(20),e(20),max,min,avg,sfitnes
CHARACTER*1 ch
LOGICAL*1 chrom1(100),chrom2(100),parent1(100),parent2(100),
# child1(100),child2(100)
STRUCTURE /population/
LOGICAL*1 chrom(100)
REAL*8 fitnes
END STRUCTURE
RECORD /population/ oldpop(1000),newpop(1000)
COMMON /ACOM/xd,dr,e,/BCOM/max,min,avg,sfitnes,mx,mn,
# /CCOM/icode,l,lchrom
Program completed on Dec '98.

c Program completed on Dec '98.
c
call GETTIM (ihr,imin,isec,i100th)
write(*,*)'Initial time',ihr,'.',imin,'.',isec,'.',i100th
idum=-i100th
if (idum.eq.0) idum=-1
write(*,*)'seed',idum
gen=0
c initial setup
call initdata(icode,popsize,maxgen,pcross,pmutate)
write(*,*)'Is there any other problem related const.s to be
# read from input data file ?(y/n)'
READ(*,*(A))ch
if (ch.eq.'y') call input()
call GETTIM (ihr,imin,isec,i100th)
call substrln()
call inisetup(idum,oldpop,popsize,lchrom,chrom1,chrom2,x)
igen=0
3  do 5 gen=igen+1,maxgen
call generate(parent1,parent2,child1,child2,chrom2,oldpop
# ,newpop,popsize,idum,lchrom,sfitnes,pcross,pmutate,x)
do 10 i=1,popsize
do 15 j=1,lchrom
oldpop(i).chrom(j)=newpop(i).chrom(j)
15 continue
oldpop(i).fitnes=newpop(i).fitnes
10 continue
call stats(popsize,oldpop)
call outdata(oldpop,gen,popsize,lchrom)
call GETTIM (ihr,imin,isec,i1100th)
if (ihr.ne.ihr) then
icmin=60-imin+imin
else
icmin=imin-imin
233
write(*,*)'Program execution time ',icmin,' mins.'
write(*,*)'Want to know the problem characteristics represented
# by the chromosomes ?( All or Best fit or No) (a/b/n)'
read(*,'(A)')ch
if (ch.eq.'a') then
  do 17 j=1,popsize
    if(j.eq.mx)write(*,*)'Best fit chromosome'
    do 20 i=1,lchrom
      chrom2(i)=oldpop(j).chrom(i)
    20 continue
  call applchar(chrom2,chrom1,x)
  continue
else if (ch.eq.'b') then
  do 25 i=1,lchrom
    chrom2(i)=oldpop(mx).chrom(i)
  25 continue
  call applchar(chrom2,chrom1,x)
endif
write(*,*)'Do you want to have a few more iterations ?(y/n)'
read(*,'(A)')ch
if (ch.eq.'n')go to 35
write(*,*)'Total # of iterations'
read(*,*)ngen
igen=maxgen
maxgen=ngen+maxgen
35 end

SUBROUTINE initdata(icode,popsize,maxgen,pcross,pmutate)
  c SGA data entry
  INTEGER popsize
  REAL*8 xd(20),dr(20),e(20)
  CHARACTER*1 a
  COMMON /ACOM/xd,dr,e
  write(*,*)'Do you want to create the input data file?(Y/N)'
  read(*,'(A)')a
  if ((a.eq.'Y').or.(a.eq.'y')) then
    OPEN(19,file=' ',err=99, status='new')
    write(*,'(A)')'No. of independent variables to undergo coding'
    read(*,*)icode
    write(19,*)(xd(i), i=1,icode)
    write(19,*)(e(i),i=1,icode)
    write(19,*)(e(i),i=1,icode)
write(*,*)'Maximum acceptable change in the variables'
read(*,*)(dr(i),i=1,icode)
write(19,*)(dr(i),i=1,icode)
write(*,*)'Enter population size'
read(*,*)popsize
write(19,*)popsize
write(*,*)'Maximum number of generations'
read(*,*)maxgen
write(19,*)maxgen
write(*,*)'Crossover probability'
read(*,*)pcross
write(19,*)pcross
write(*,*)'Mutation probability'
read(*,*)pmutate
write(19,*)pmutate
close(19)

else

OPEN (19,file=' ',err=99,status='old')
write(*,*)'No. of independent variables to undergo coding'
read(19,*)icode
write(*,*)icode
write(*,*)'Origin'
read(19,*)(xd(i),i=1,icode)
write(*,*)(xd(i),i=1,icode)
write(*,*)'Accuracy required in each variables'
read(19,*)(e(i),i=1,icode)
write(*,*)(e(i),i=1,icode)
write(*,*)'Maximum acceptable change in the variables'
read(19,*)(dr(i),i=1,icode)
write(*,*)(dr(i),i=1,icode)
write(*,*)'Enter population size'
read(19,*)popsize
write(*,*)popsize
write(*,*)'Maximum number of generations'
read(19,*)maxgen
write(*,*)maxgen
write(*,*)'Crossover probability'
read(19,*)pcross
write(*,*)pcross
write(*,*)'Mutation probability'
read(19,*)pmutate
write(*,*)pmutate
close(19)
endif
go to 10
99 stop' isga.dat does not exist'
10 return

end

SUBROUTINE outdata(pop,gen,popsize,lchrom)
c write population report at each generation
   INTEGER gen,popsize,lchrom
   REAL*8 max,min,avg,sfitnes
   STRUCTURE /population/
      LOGICAL* 1 chrom(100)
      REAL*8 fitnes
   END STRUCTURE
   RECORD /population/ pop(1000)
   COMMON /BCOM/max,min,avg,sfitnes,mx,mn
   write(*,*)'Population report'
   write(*,*)'Current population chromosomes and fitness values'
   do 5 j=1,popsize
   c write(*,*)'chrom'j,' ',(pop(j).chrom(i),i=1,lchrom)
   c write(*,*)'fitness',pop(j).fitnes
   5 continue
   write(*,*)'generation statistics'
   write(*,*)'generation # ',gen+1
   write(*,*)'maximum fitness ',max
   write(*,*)'minimum fitness ',min
   write(*,*)'average fitness ',avg
   write(*,*)'sum fitness ',sfitnes
   write(*,*)'maximum fitness chromosome'
   write(*,*)(pop(mx).chrom(i),i=1,lchrom)
   return
end

**************************************************************************************
FILE: sgaultil.for
**************************************************************************************
This file contains following functions and subroutines:

substrlin: calculates length of the substrings corresponding to the different design
variables and also calculates the overall length of the chromosome

iniseth: calls the subroutines initpop, initrep, stats

initpop: randomly generates the initial population

initrep: writes the report for the initial population on the standard output device

generate: generates new generation of population from the old generation by applying
selection, and mutation embedded crossover operator

stats: calculates the population statistics, finds out fittest and the least fit
chromosome and the average fitness for the population

convert: converts bit strings to their corresponding decimal values

decode: converts Gray code to binary number then binary to decimal
**select:**  simulates natural selection using roulette wheel mechanism

**mutation:** simulates bit mutation operation

**crossover:** simulates single point crossover operation

**flip:** simulates flipping of an unbiased coin

**ran3:** generates sequence of random fractions, same as in APPENDIX B, to avoid repetition this function is excluded from the following code listing.

**fitfunc:** calculates fitness for a chromosome

```fortran
SUBROUTINE substrln()
  c determines bit string length for each of the variables and also
  c for the chromosome
  INTEGER i,l(10),lchrom, icode
  REAL*8 dr(20),e(20),xd(20)
  COMMON /ACOM/xd,dr,e,/CCOM/icode,l,lchrom
  lchrom=0
  do 5 i=1, icode
    l(i)=idnint(dlog10(dr(i)/e(i))/dlog10(2.0))
    lchrom=lchrom+l(i)
    c write(*,*)T,l(i)
  5 continue
  c write(*,*)ichrom',lchrom
  c pause
  return
end

SUBROUTINE inisetup(idum,pop,popsize,lchrom,
&  % chrom1,chrom2,x)
  INTEGER popsize
  REAL*8 x(20),max,min,avg,sfitnes
  LOGICAL*! chrom1(100),chrom2(100)
  STRUCTURE /population/
    LOGICAL* 1 chrom(100)
  REAL*8 fitnes
  END STRUCTURE
  RECORD /population/ pop(1000)
  COMMON /BCOM/ max,min,avg,sfitnes,mx,mn
  c initialize GA parameters
    call initpop(popsize,lchrom,pop,idum,chrom1,chrom2,x)
    call stats(popsize,pop)
    call initrep(lchrom)
```
SUBROUTINE initpop(popsize,lchrom,pop,idum,chrom1,chrom2,x)
c initialize a population at random
INTEGER i,il,popsize, lchrom
REAL*8 x(20),fitfunc
LOGICAL*1 chrom1(100),chrom2(100),flip
STRUCTURE /population/
LOGICAL*1 chrom(100)
REAL*8 fitnes
END STRUCTURE
RECORD /population/ pop(1000)
do 5 i=1,popsize
do 10 il=1,lchrom
pop(i).chrom(il)=flip(0.5, idum)
chrom1(il)=pop(i).chrom(il)
10 continue
write(*,*)'chrom ',(pop(i).chrom(il),il=1,lchrom)
pause
pop(i).fitnes=fitfunc(chrom1,chrom2,x)
write(*,*)'fitness',pop(i).fitnes
pause
5 continue
return
end

SUBROUTINE initrep(lchrom)
c initial values of some of the GA parameters
REAL*8 max,min,avg,sfitnes
COMMON /BCOM/ max,min,avg,sfitnes,mx,mn
write(*,*)'chromosome length ',lchrom
write(*,*)'First generation statistics'
write(*,*)'maximum fitness ',max
write(*,*)'minimum fitness ',min
write(*,*)'sum of fitness ',sfitnes
write(*,*)'average fitness ',avg
pause
return
end

SUBROUTINE generate(parent1,parent2,child1,child2,chrom2,pop*
*,newpop,popsize,idum,lchrom,sfitnes,pcross,pmutate,x)
c create a new generation thru select,crossover and mutation
c generate assumes an even-numbered popsize
INTEGER mate1,mate2,j,jcross,popsize,select
REAL*8 x(20),sfitnes,fitfunc
LOGICAL*1 parent1(100),parent2(100),child1(100),child2(100), chrom2(100)

STRUCTURE /population/
LOGICAL*1 chrom(100)
REAL*8 fitnes
END STRUCTURE

RECORD /population/ newpop(1000),pop(1000)

c select,crossover and mutation until newpop is filled
do 5 i=1,popsize,2

c pick pair of mates for mating pool
matel=select(popsize,sfitnes,pop)
mate2=select(popsize,sfitnes,pop)
c write(*,*)'matel ,mate2',matel ,mate2
c pause
c crossover and mutation - mutation embedded with crossover
do 10 j=1,lchrom
parent1(j)=pop(matel ).chrom(j)
parent2(j)=pop(mate2).chrom(j)
10 continue
call crossover(parent1,parent2,child1,child2,lchrom,
@ pcross,pmutate,idum)
do 15 j=1,lchrom
newpop(i).chrom(j)=child1(j)
newpop(i+1).chrom(j)=child2(j)
15 continue
c write(*,*)'newpop ,i,(newpop(i).chrom(j)j=1,lchrom)
c write(*,*)'newpop ,i+1,(newpop(i+1).chrom(j)j=1,lchrom)
c decode string, evaluate fitness values
newpop(i).fitnes=fitfunc(child1,chrom2,x)
newpop(i+1).fitnes=fitfunc(child2,chrom2,x)
c write(*,*)'fitnes ,i,newpop(i).fitnes
c write(*,*)'fitnes ,i+1,newpop(i+1).fitnes
5 continue
c pause
c return
c end

SUBROUTINE stats (popsize,pop)
c calculate population statistics
INTEGER j,popsize
REAL*8 max,min,avg,sfitnes,fitnes1
STRUCTURE /population/
LOGICAL*1 chrom(100)
REAL*8 fitnes
END STRUCTURE
RECORD /population/ pop(1000)
COMMON /BCOM/ max,min,avg,sfitnes,mx,mn

c initialize
sfitnes=pop(1).fitnes
min=pop(1).fitnes
max=pop(1).fitnes

! loop for max, min, sfitnes
do 5 j=2,popsize
    fitnes1=pop(j).fitnes

! accumulate fitness sum
    sfitnes=sfitnes+fitnes1
    if(fitnes1.gt.max) then
        max=fitnes1
        mx=j
    endif
    if(fitnes1.lt.min) then
        min=fitnes1
        mn=j
    endif
5 continue

! calculate average
    avg=sfitnes/popsise

return

SUBROUTINE convert(chrom1,chrom2,x)
! converts sub bit strings to their corresponding decimal values
INTEGER i, i1, i2, icode, l(10), Ichrom
REAL*8 x(20),xd(20),dr(20),e(20),decode
LOGICAL*1 chrom1(100),chrom2(100)
COMMON /ACOM/xd,dr,e;/CCOM/icode,l, Ichrom

i2=1
do 10 i1=1,icode
    do 5 i=1,l(i1)
        chrom2(i)=chrom1(i1)
    5 continue
    write(*,*)'chroml ',chroml(i2)
    write(*,*)'chrom2 ',chrom2 ,i
    i2=i2+1
10 continue

x(i1)=xd(i1)+dr(i1)*decode(chrom2,l(i1))/((2.**l(i1))-1.)
write(*,*)x,dr,decode,x(i1),dr(i1),decode(chrom2,l(i1))
pause
return

FUNCTION decode (chroml,lbits)
! converts Gray code to binary number then binary to decimal
REAL*8 accum,powerof2,decode
LOGICAL*1 chrom1(100),chrom2(100)
c Gray to binary conversion
c start from MSB
c write(*,*)'chrom1after ',(chrom1(i),i=1,lbits)
c chrom2(lbits)=chrom1(lbits)
do 5 i=1,lbits-1
   chrom2(lbits-i)=chrom2(lbits-i+1).XOR.chrom1(lbits-i)
 5 continue
c write(*,*)'chrom2after ',(chrom2(i),i=1,lbits)
c pause
c binary to decimal
   accum=0.0
   powerof2=1.0
   do 10 i=1,lbits
      if (chrom2(i)) then
         accum=accum+powerof2
      endif
      powerof2=powerof2*2
 10 continue
   decode=accum
c write(*,*)'decode',decode
end

FUNCTION select (popsize,sfitnes,pop)
c Simulates natural selection using spins of a roulette wheel

   INTEGER popsize,j,select
   REAL*8 rand,partsum,sfitnes
   STRUCTURE /population/
      LOGICAL*1 chrom(100)
      REAL*8 fitnes
   END STRUCTURE
   RECORD /population/ pop(1000)
      partsum=0.
      j=0
      rand= ran3(idum)*sfitnes
   c find wheel slot
   do 5 j=1,popsize
      partsum=partsum+pop(j).fitnes
      if (partsum.ge.rand) go to 6
   5 continue
   c return individual no.
   6 select=j
      return
end

241
FUNCTION mutation (allelval, pmutate, idum)
    c mutate an allele with prob pmutate, count no. of mutations

    LOGICAL*1 mutation, allelval, mutate, flip
    REAL pmutate
    c flip the biased coin with pmutate
    mutate = flip(pmutate, idum)
    if (mutate) then
        c nmutate = nmutate + 1
        c change bit value
        mutation = .not. allelval
    else
        c no change
        mutation = allelval
    endif
    return
end

SUBROUTINE crossover (parent1, parent2, child1, child2, lchrom, ! pcross, pmutate, idum)
    c cross 2 parent strings, place in 2 child strings

    REAL ran3
    LOGICAL*1 parent1(100), parent2(100), child1(100), child2(100), ! mutation, flip
    c do crossover with pcross
    if (flip (pcross, idum)) then
        c choice of crossover site, jcross = random integer
        jcross = nint(1.0 + ran3(idum) * (lchrom - 1))
        c increment crossover counter
        ncross = ncross + 1
    else
        c otherwise set cross-site to force mutation
        jcross = lchrom
    endif
    c write(*, *) 'jcross', jcross
    c 1st exchange, 1 to 1 and 2 to 2
    do 5 j = 1, jcross
        c write(*, *) 'gene # ', j
        child1(j) = mutation(parent1(j), pmutate, idum)
        child2(j) = mutation(parent2(j), pmutate, idum)
    5 continue
    c write(*, *) 'child1 ', (child1(j), j = 1, jcross)
    c write(*, *) 'child2 ', (child2(j), j = 1, jcross)
    c pause
    c 2nd exchange, 1 to 2, 2 to 1
    if (jcross.ne.lchrom) then
        do 10 j = jcross + 1, lchrom
        10 continue
child1(j) = mutation(parent2(j), pmutate, idum)
child2(j) = mutation(parent1(j), pmutate, idum)

10 continue

c write(*,*) child1 ', (child1(j), j=1+jcross, lchrom)
c write(*,*) child2 ', (child2(j), j=1+jcross, lchrom)
c pause
endif
return
end

FUNCTION flip (prob, jdum)

c Uses ran3

c A biased coin is flipped - true if heads
   REAL ran3, prob
   LOGICAL*1 flip
   DATA jdum/1/
c use of DATA statement instead of assignment statement for
   initialization of jdum is important. This allows subsequent
   changes in jdum during a single run of the main program.
   
   if(prob.eq.1.0) then
     flip=.true.
   else
     rand=ran3(jdum)
     flip=(rand.lt.prob)
   endif
   return
end

FUNCTION fitfunc (chrom1, chrom2, x)

c to calculate fitness value of a chromosome
   REAL*8 meritf, fitfunc, x(20)
   LOGICAL*1 chrom1(100), chrom2(100)
   CHARACTER*100 charstr
   COMMON /ECOM/meritf, charstr
   call convert(chrom1, chrom2, x)
   call objfunc(x)
   
c write(*,*) meritf, meriff
   fitfunc=1./(1.+meritf)
   return
end
FORTRAN CODE FOR CALCULATING COOKE TRIPLET MERIT FUNCTIONS

The codes (written in Microsoft FORTRAN77 version 5.0) for the triplet support file triplsup.for and the file for calculating triplet merit functions mftripl.for are listed here.

******************************************************************************
FILE: triplsup.for
******************************************************************************

The file is developed as a support file for structural design of Cooke triplet problem. Works as an interface between the SGA files (sgamain.for & sgautil.for) and thin Cooke triplet merit function mftripl.for. The gsa files (gsamain.for & gsautil.for) work with the same triplet files with slight modifications in the common list of variables.

This file contains following functions and subroutines:

applchar : to write application specific information. Here it writes Cooke triplet specific data

gconvert: formulation of thin Cooke triplet requires the use of glass variables. In our problem they are selected from a wellchosen list of 64 glasses contained in glaslib.dat file. This subroutine converts the decimal equivalents of GRAY coded glass variables to the nearest integers possible and the corresponding glass refractive index and dispersion values are read from the glaslib.dat file (Appendix E).

input : to read problem (thin Cooke triplet) specific data interactively from the console and write it to a file or from an existing file named isgatpl.for.

sysconst : calculates the constants to be applied to the normalized forms of primary aberrations to get the actual Seidel aberration values. These constants are calculated using the paraxial specifications for each of the singlet, obtained by raytracing.

objfunc : This subroutine works as a mediator between the mfunc subroutine in mftripl.for file & fitfunc subroutine in sgautil.for file.

C   wl: wavelength; HH: Lagrange invariant for the system; h(10): height of
C   p.m.r. at different surfaces; hbar(10): height of p.p.r. at different
C   surfaces;x(20): running variable (9 constructional parameters, 3 for of
C   the comp.(shape variable, glass r.i. & dispersion) of 3 primary
C   aberration values for 3 comp.s; sat(6): overall primary aberration
C   targets; f(5): normalized aberrations and pseudoaberrations
C   q(9): normalizing factors for 3 comp.s; y(3): conjugate
C   variables for each of the components;
C   Development started sometime in December 1998.

SUBROUTINE applchar(chrom1,chrom2,x)
REAL*8 x(20),meritf,xa(20),sat(6),y(3),q(9),ks,hs,wl,HH,h(10),
fitnes= fitfunc (chrom1,chrom2,x)
xa(1)=x(i)
xa(2)=x(i+n)
xa(3)=x(i+2*n)
call aputra(xa,aur,curvatur)
write(*,*)'Surface curvatures ',curvatur(1),curvatur(2)
write(*,*)'Aperture Utilization Ratio ',aur
endif
45 continue
call GETTIM (ihr,imin,isec,i100th)
write(*,*)'Final time',ihr,imin,isec,i100th
29 end

SUBROUTINE gconvert(x)
c  glasses are picked from a list of 64 preferred glasses
INTEGER l(10)
REAL*8 x(20),ri(3),deln(3 ),y(3 ),q(9)
COMMON /CCOM/ icode,l,lchrom,/FCOM/n,id,iosa,y,q
do 5 i=l,n
  ip=idnint(x(icode+i-n))
  OPEN(20,file='glaslib.dat',recl= 17,form-FORMATATTED')
c write(*,*)'ip'.ip
  doj=1,ip-l
    read(20,*)
c continue
  read(20,'(BN, f8.6, lx,f8.6)')ri(i),deln(i)
close(20)
c write(*,*)'ri,deln',ri(i),deln(i)
5 continue
do 10 i=l,n
  x(id-2*n+i)=ri(i)
  x(id-n+i)=deln(i)
10 continue
c write(*,*),x,,(x(i),i=l,id)
return
end

SUBROUTINE input()
c  input data
REAL*8 pow(10),w(6),sat(6),HH,h(10),hbar(10),y(3),q(9)
  ,u(10),c(10),ri(3)
CHARACTER* 1 a
COMMON /GCOM/ wIHH,h,hbar,pow,/DCOM/ w,sat,/FCOM/n,id,iosa,y,q
  /LCOM7 u,c,ri
write(*,*),DInput for thin Cooke triplet problem'
c write(*,*)'Do you want to create the input data file?(Y/N)'
  read(*,'(A)')a
if ((a.eq.'Y',).or.(a.eq.'y')) then
  open(19,file- status-new')
c write(*,*)'Do you want to see optimum splitting of aber.s
246
1 among the comp.s ? (0 for no)
read(*,*)iosa
write(19,*),iosa
write(*,*)' # components'
read(*,*)n
write(19,*),n
write(*,*)' Weights for the merit function'
read(*,*)(w(i),i=1,6)
write(19,*),w(i),i=1,6
write(*,*)' System constants : Lagrange invariant'
read(*,*)HH
write(19,*),HH
write(*,*)' Operating wavelength'
read(*,*)wl
write(19,*),wl
write(*,*)' Constants for each of the components:'
write(*,*)' Height of pmt'
read(*,*)(h(i),i=1,n)
write(19,*),h(i),i=1,n
write(*,*)' Height of ppr'
read(*,*)(hbar(i),i=1,n)
write(19,*),hbar(i),i=1,n
write(*,*)' Convergence angles due to pmt'
read(*,*)(u(i),i=1,n+1)
write(19,*),u(i),i=1,n+1
write(*,*)' Surface curvatures'
read(*,*)(c(i),i=1,2*n)
write(19,*),c(i),i=1,2*n
write(*,*)' Refractive indices'
read(*,*)(ri(i),i=1,n)
write(19,*),ri(i),i=1,n
write(*,*)' Aberration targets (sph. abr.(SI), coma(SII),
@ astig.(SIII),dist.(SV), long.chr.abr.(CL), trans.chr.abr.(CT))'
read(*,*)(sat(i),i=1,6)
write(19,*),sat(i),i=1,6
close(19)
else
open(19,file=' ',err=99,status='old')
write(*,*)' Do you want to see optimum splitting of aber.s
1 among the comp.s ? (0 for no)
read(*,*)iosa
write(*,*)iosa
write(*,*)' # components'
read(*,*)n
write(*,*)n
write(*,*)' Weights for the merit function'
read(19,*),w(i),i=1,6
write(*,*)w(i),i=1,6
write(*,*)' System constants : Lagrange invariant'
read(19,*),HH
write(:,:,*)HH
write(*,*)HH
write(*,*)'Operating wavelength'
read(19,*)wl
write(*,*)wl
write(*,*)'Constants for each of the components:'
write(*,*)'Height of pmr'
read(19,*)h(i),i=1,n
write(*,*)h(i),i=1,n
write(*,*)'Height of ppr'
read(19,*)hbar(i),i=1,n
write(*,*)hbar(i),i=1,n
write(*,*)'Convergence angles due to pmr'
read(19,*)u(i),i=1,n+1
write(*,*)u(i),i=1,n+1
write(*,*)'Surface curvatures'
read(19,*)c(i),i=1,2*n
write(*,*)c(i),i=1,2*n
write(*,*)'Refractive indices'
read(19,*)ri(i),i=1,n
write(*,*)ri(i),i=1,n
write(*,*)'Aberration targets (sph. abr.(SI), coma(SII),
astig.(SIII),dist.(SV), long.chr.abr(CL), trans.chr.abr.(CT))'
read(19,*)sat(i),i=1,6
write(*,*)sat(i),i=1,6
close(19)
endif

go to 10
99 stop 'isgatpl.dat does not exist'
10 call sysconst()
return
end

SUBROUTINE sysconst()
calculation of system constants

REAL*8 pow(10),wl,HH,y(3),h(10),hbar(10),q(9),u(10),c(10),ri(3)
COMMON /GCOM/ wl,HH,h,hbar,pow,/FCOM/n,id,iosa,y,q,/LCOM/u,c,ri
id=9
in=1
do 5 i=1,n
   y(i)=(u(i+1)+u(i))/((u(i+1)-u(i))
pow(i)=(ri(i)-1.)*(c(in)-c(in+1))
in=in+2
   q(i)=(h(i)*4)*(pow(i)**3)/(wl)
p(i)=HH*(h(i)**2)*(pow(i)**2)/(wl)
p(i+2*n)=(h(i)**2)*pow(i)/(wl)
c   write(*,*)y,pow,q,y(i),pow(i),q(i),q(i+n),q(i+2*n)
5   continue
   call mf_const()
return
SUBROUTINE objfunc(x)
c works as mediator between functions MFUNC and FITFUNC
REAL*8 x(20),y(3),q(9)
COMMON /FCOM/n,id,iosa,y,q
if(iosa.eq.0) then
  call gconvert (x)
endif
  call mfunc(x)
return
end

FILE: mftripl.for

This file contains the following functions and subroutines:

phi1: calculates the triplet merit function due to the monochromatic primary aberrations
phi2: calculates the triplet merit function due to longitudinal and transverse chromatic aberrations
mf_const: calculates the constants related to the merit function
mfunc: calculates the overall triplet merit function
aputra: this subroutine calculates maximum aperture utilization ratio
abers: calculates the normalized forms of primary central aberrations for a single thin lens

c Calculates meritfunction for thin Cooke triplet using stop shift
c formulae and the Seidel Coefficients for primary spherical aberration, central coma, astigmatism and distortion for each of the c components. Ref."Structural Design of multicomponent lens systems",
c variables are same as explained in triplsup.for file.
c w(i): weights for different parts of meritfunction

FUNCTION phi1()
c Formulation of merit function for multicomponent lens system,
c considering the Seidel Coefficients for primary spherical c aberration, central coma, astigmatism and distortion.

C CALCULATIONS RELATED TO THE MERIT FUNCTION,WHICH ARE TO BE C REPEATED FREQUENTLY.
REAL*8 a(5,10),b(6),sum1,sat(6),sum,phi1,sei_abr(9),q(9)
1 ,h(10),hbar(10),pow(10),HH,w1,w(6),wd(4),sum_abr(6),sumd,y(3)
COMMON /DCOM/w,sat,/GCOM/w1,HH,h,hbar,pow,/HCOM/a,b,/ICOM/
1 sum_abr,sei_abr,/FCOM/n,id,iosa,y,q
  do 5 i=1,4
    wd(i)=w(i)
  5 continue
FUNCTION phi2()

      c  Formulation of merit function for multicomponent lens system,
      c  considering the primary longitudinal chromatic aberration,
      c  and transverse chromatic aberration.
      C  CALCULATES MERIT FUNCTION, CALCULATIONS TO BE
      C  REPEATED FREQUENTLY.
      REAL*8 a(5,10),b(6),sat(6),HH,wl,sum,sei_abr(9),
      1 h(10),hbar(10),pow(10),w(6),sum_abr(6),wd(2),phi2,y(3),q(9)
      COMMON /DCOM/w,sat/COM/wl,HH,h,hbar,pow,/HCOM/a,b,/ICOM/
      1 sum_abr,sei_abr,/COM/n,id,iosa,y,q

      do 5 i=1,2
         wd(i)=w(4+i)
      5 continue
      sum=0.
      do 7 j=1,n
         sum=sum+a(2,j)*sei_abr(j+2*n)*wl
      7 continue
      phi2 = 0.
      do 10 i=1,2
         sum1=0.
      10 continue
      phi2 = phi2+(wd(i)*(sum1+b(i+4)))**2
SUBROUTINE mf_const()
C CALCULATIONS RELATED TO THE CONSTANTS OF MERIT FUNCTION.
REAL*8 pow_sum,HH,wi,ps,sum_abr(6),y(3),q(9),sei_abr(9)
1 ,sat(6),h(10),hbar(10),pow(10),a(5,10),b(6),w(6)
COMMON /DCOM/w,sat,/GCOM/wl,HH,h,hbar,pow,/HCOM/a,b,/FCOM/n,id,
1iosa,y,q,/ICOM/sum_abr,sei_abr
ps=0.63
do 2 i=T,2
b(i)=-sat(i)
b(i+4)=-sat(i+4)
2 continue
sum=0.
pow_sum=0.
do 5 i=1,n
pow_sum=pow_sum+pow(i)
sum= sum + (hbar(i)/h(i))*pow(i)
a(1,i)=1.
a(1,n+i)=0.
a(2,i)=hbar(i)/h(i)
a(2,i+n)=1.
a(3,i)=(hbar(i)/h(i))**2
a(3,i+n)=2.*(hbar(i)/h(i))
a(4,i)=(hbar(i)/h(i))**3
a(4,i+n)=3.*(hbar(i)/h(i))**2
5 continue
b(3)=-(sat(3)-(HH**2)*pow_sum/wi)
b(4)=-sat(4)+(3.+ps)*(HH**2)*sum/wl
return
end

SUBROUTINE APUTRA(X,aur,cd)
C THIS SUBROUTINE CALCULATES MAXIMUM APERTURE UTILIZATION RATIO
REAL*8 X(20),ks,hs,CD(6),cdd,aur
COMMON /KCOM/ks,hs
CD(1)=Ks*(X(1)+1.)/(2.*(X(2)-1.))
CD(2)=Ks*(X(1)-1.)/(2.*(X(2)-1.))
c write(*,*)'Surface curvatures',(cd(i),i=1,2)
c DDD=ABS(CD(1))
IF (CDD.GE.ABS(CD(2))) GO TO 7
CDD=ABS(CD(2))
GO TO 10
7 CDD=ABS(CD(1))
SUBROUTINE ABERS(X)
C This subroutine calculates the primary spherical aberration,
C central coma, and chromatic aberration for a single thin lens
REAL*8 S1,S2,CL,X(20),f(5),A1,B1,C1,D1,E1,G1,ys
COMMON /JCOM/f,ys

A1=(X(2)**2-1.)/(X(2)+2.)
B1=2.*(X(2)**2-1.)/(X(2)+2.)
C1=(X(2)**2)/(X(2)-1.)
D1=X(2)/(X(2)+2.)
E1=(X(2)+1.)/(X(2)*(X(2)-1.))
G1=(2.*X(2)+1.)/X(2)
S1=(1./4.)*(A1*(X(1)-B1*ys)**2)+C1-D1*(ys**2)
S2=(1./2.)*(E1*X(1)-G1*ys)
CL=X(3)/(X(2)-1.)
F(1)=S1
F(2)=S2
F(3)=CL
RETURN
END

SUBROUTINE mfunc(X)
C CALCULATIONS RELATED TO THE MERIT FUNCTION, WHICH ARE TO BE
C REPEATED FREQUENTLY.
REAL*8 X(20),MERITF,x(20),f(5),ys,sei_abr(9),q(9),phi1,phi2
1,sum_abr(6),y(3)
CHARACTER*100 charstr
COMMON /ECOM/ meritf,charstr,/FCOM/n,id,iosa,y,q,/ICOM/sum_abr,
1,sei_abr,/JCOM/f,ys
charstr = 'OPTICAL MERIT FUNCTION FOR THIN COOKE TRIPLET'
if (iosa.eq.0) then
  do 5 i=1,n
    ys=y(i)
    XA(1)=X(i)
    x(2)=x(i+n)
    XA(3)=x(i+2*n)
    CALL ABERS(XA)
    c write(*,*),f(i),j=1,3
    sei_abr(i)=(f(1)*Q(i))
    sei_abr(i+n)=(f(2)*Q(i+n))
    sei_abr(i+2*n)=(f(3)*Q(i+2*n))
  c write(*,*),sei_abr,sei_abr(i),sei_abr(i+n),sei_abr(i+2*n)
  c pause
  5 continue
else
  do 10 i=1,n
    sei_abr(i)= X(i)
10 continue

sei_abr(i+n) = x(i+n)
sei_abr(i+2*n) = x(i+2*n)
cwrite(*,*)'sei_abr', sei_abr(i), sei_abr(i+n), sei_abr(i+2*n)
cpause
10 continue
endif
MERITF = phi1() + phi2()
cwrite(*,*)'phi1,phi2,meritf', phi1(), phi2(), meritf
cpause
RETURN
END
Sixty-five SCHOTT Optical Glasses are listed below with their $n_d$ (refractive index at Helium d line) and dispersion ($\delta n = n_F - n_C$) values.

F - blue hydrogen line, 486.13 nm  
C - red hydrogen line, 656.27 nm  
d - yellow helium line, 587.56 nm

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Glass types</th>
<th>$n_d$</th>
<th>$\delta n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BaK1</td>
<td>1.572500</td>
<td>0.009948</td>
</tr>
<tr>
<td>2</td>
<td>BaK 2</td>
<td>1.539960</td>
<td>0.009043</td>
</tr>
<tr>
<td>3</td>
<td>BaK 4</td>
<td>1.568830</td>
<td>0.010135</td>
</tr>
<tr>
<td>4</td>
<td>BaK 5</td>
<td>1.556710</td>
<td>0.009492</td>
</tr>
<tr>
<td>5</td>
<td>K 3</td>
<td>1.518230</td>
<td>0.008787</td>
</tr>
<tr>
<td>6</td>
<td>K 5</td>
<td>1.522489</td>
<td>0.008784</td>
</tr>
<tr>
<td>7</td>
<td>K 7</td>
<td>1.511121</td>
<td>0.008461</td>
</tr>
<tr>
<td>8</td>
<td>K 10</td>
<td>1.501371</td>
<td>0.008888</td>
</tr>
<tr>
<td>9</td>
<td>SK 2</td>
<td>1.607381</td>
<td>0.010721</td>
</tr>
<tr>
<td>10</td>
<td>SK 3</td>
<td>1.608810</td>
<td>0.010332</td>
</tr>
<tr>
<td>11</td>
<td>SK 4</td>
<td>1.612720</td>
<td>0.010451</td>
</tr>
<tr>
<td>12</td>
<td>SK 5</td>
<td>1.589130</td>
<td>0.009615</td>
</tr>
<tr>
<td>13</td>
<td>SK 6</td>
<td>1.613750</td>
<td>0.010882</td>
</tr>
<tr>
<td>14</td>
<td>SK 7</td>
<td>1.607290</td>
<td>0.010214</td>
</tr>
<tr>
<td>15</td>
<td>SK 8</td>
<td>1.611171</td>
<td>0.010929</td>
</tr>
<tr>
<td>16</td>
<td>SK 10</td>
<td>1.622801</td>
<td>0.010945</td>
</tr>
<tr>
<td>17</td>
<td>SK 14</td>
<td>1.603110</td>
<td>0.009952</td>
</tr>
<tr>
<td>18</td>
<td>SK 15</td>
<td>1.622990</td>
<td>0.010731</td>
</tr>
<tr>
<td>19</td>
<td>SK 16</td>
<td>1.620411</td>
<td>0.010284</td>
</tr>
<tr>
<td>20</td>
<td>SK N18</td>
<td>1.638541</td>
<td>0.011521</td>
</tr>
<tr>
<td>21</td>
<td>LaK N6</td>
<td>1.642499</td>
<td>0.011085</td>
</tr>
<tr>
<td>22</td>
<td>LaK N7</td>
<td>1.651601</td>
<td>0.011134</td>
</tr>
<tr>
<td>23</td>
<td>LaK 8</td>
<td>1.713003</td>
<td>0.013245</td>
</tr>
<tr>
<td>24</td>
<td>LaK N9</td>
<td>1.691002</td>
<td>0.012631</td>
</tr>
<tr>
<td>25</td>
<td>LaK 10</td>
<td>1.719999</td>
<td>0.014282</td>
</tr>
<tr>
<td>26</td>
<td>LaK N12</td>
<td>1.677900</td>
<td>0.012280</td>
</tr>
<tr>
<td>27</td>
<td>LaK N13</td>
<td>1.693501</td>
<td>0.013004</td>
</tr>
<tr>
<td>28</td>
<td>LaK 21</td>
<td>1.640496</td>
<td>0.010658</td>
</tr>
<tr>
<td>29</td>
<td>LaK 24</td>
<td>1.696800</td>
<td>0.012404</td>
</tr>
<tr>
<td>30</td>
<td>SSK 2</td>
<td>1.622301</td>
<td>0.011707</td>
</tr>
<tr>
<td>31</td>
<td>SSK N5</td>
<td>1.658440</td>
<td>0.012940</td>
</tr>
<tr>
<td>32</td>
<td>SSK N8</td>
<td>1.617720</td>
<td>0.012412</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>33</td>
<td>KF 6</td>
<td>1.517421</td>
<td>0.009913</td>
</tr>
<tr>
<td>34</td>
<td>KF 9</td>
<td>1.523412</td>
<td>0.010166</td>
</tr>
<tr>
<td>35</td>
<td>BaLF 4</td>
<td>1.579569</td>
<td>0.010790</td>
</tr>
<tr>
<td>36</td>
<td>BaLF 5</td>
<td>1.547391</td>
<td>0.010207</td>
</tr>
<tr>
<td>37</td>
<td>LLF 1</td>
<td>1.548140</td>
<td>0.011980</td>
</tr>
<tr>
<td>38</td>
<td>LLF 6</td>
<td>1.531719</td>
<td>0.010905</td>
</tr>
<tr>
<td>39</td>
<td>BaF 3</td>
<td>1.582670</td>
<td>0.012538</td>
</tr>
<tr>
<td>40</td>
<td>BaF 4</td>
<td>1.605620</td>
<td>0.013787</td>
</tr>
<tr>
<td>41</td>
<td>BaF 8</td>
<td>1.623740</td>
<td>0.013270</td>
</tr>
<tr>
<td>42</td>
<td>BaF N10</td>
<td>1.670031</td>
<td>0.014222</td>
</tr>
<tr>
<td>43</td>
<td>BaF N11</td>
<td>1.666721</td>
<td>0.013769</td>
</tr>
<tr>
<td>44</td>
<td>BaF 13</td>
<td>1.668920</td>
<td>0.014877</td>
</tr>
<tr>
<td>45</td>
<td>BaF 50</td>
<td>1.682726</td>
<td>0.015341</td>
</tr>
<tr>
<td>46</td>
<td>BaF 51</td>
<td>1.652242</td>
<td>0.014517</td>
</tr>
<tr>
<td>47</td>
<td>BaF 52</td>
<td>1.608589</td>
<td>0.013104</td>
</tr>
<tr>
<td>48</td>
<td>LF 5</td>
<td>1.581440</td>
<td>0.014233</td>
</tr>
<tr>
<td>49</td>
<td>LF 7</td>
<td>1.575011</td>
<td>0.013860</td>
</tr>
<tr>
<td>50</td>
<td>BaSF 1</td>
<td>1.626061</td>
<td>0.016068</td>
</tr>
<tr>
<td>51</td>
<td>BaSF 2</td>
<td>1.664460</td>
<td>0.018545</td>
</tr>
<tr>
<td>52</td>
<td>BaSF 10</td>
<td>1.650160</td>
<td>0.016608</td>
</tr>
<tr>
<td>53</td>
<td>BaSF 51</td>
<td>1.723728</td>
<td>0.018991</td>
</tr>
<tr>
<td>54</td>
<td>BaSF 52</td>
<td>1.701810</td>
<td>0.017112</td>
</tr>
<tr>
<td>55</td>
<td>F 1</td>
<td>1.625882</td>
<td>0.017530</td>
</tr>
<tr>
<td>56</td>
<td>F 2</td>
<td>1.620040</td>
<td>0.017050</td>
</tr>
<tr>
<td>57</td>
<td>F 4</td>
<td>1.616591</td>
<td>0.016834</td>
</tr>
<tr>
<td>58</td>
<td>F 5</td>
<td>1.603420</td>
<td>0.015868</td>
</tr>
<tr>
<td>59</td>
<td>F 6</td>
<td>1.636359</td>
<td>0.018005</td>
</tr>
<tr>
<td>60</td>
<td>F 7</td>
<td>1.625361</td>
<td>0.017586</td>
</tr>
<tr>
<td>61</td>
<td>F 8</td>
<td>1.595510</td>
<td>0.015200</td>
</tr>
<tr>
<td>62</td>
<td>TiF N5</td>
<td>1.593555</td>
<td>0.016713</td>
</tr>
<tr>
<td>63</td>
<td>SF 1</td>
<td>1.717360</td>
<td>0.024307</td>
</tr>
<tr>
<td>64</td>
<td>SF 2</td>
<td>1.647689</td>
<td>0.019135</td>
</tr>
<tr>
<td>65</td>
<td>SF 3</td>
<td>1.740002</td>
<td>0.026244</td>
</tr>
</tbody>
</table>
LIST OF PUBLICATIONS BY THE AUTHOR


